Analiza primjenjivosti alata za ispitivanje sigurnosti programske podrške

Javorek, Kristina

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski sveučilišni studij Računarstvo

ANALIZA PRIMJENJIVOSTI ALATA ZA ISPITIVANJE SIGURNOSTI PROGRAMSKE PODRŠKE

Diplomski rad

Kristina Javorek

Osijek, 2018

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 08.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Kristina Javorek
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 845 R, 19.09.2017.
OIB studenta:	27868382867
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	Verena Zaiser, M.Sc.
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva:	Doc.dr.sc. Josip Balen
Naslov diplomskog rada:	Analiza primjenjivosti alata za ispitivanje sigurnosti programske podrške
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U diplomskom radu potrebno je istražiti i analizirati raspoložive alate za ispitivanje sigurnosti programske podrške, kao i tehnike detektiranje i oblike ranjivosti programskog koda. Nadalje, prikladne alate potrebno je implementirati kao klijentske alate ili usluge u GitLabu, te ih primijeniti na odgovarajućem programskom projektu web aplikacije i dati procjenu ranjivosti s preporukama za poboljšanje programskog koda. (sumentorica: Verena Zaiser, M.Sc., Aaronprojects, Njemačka)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	08.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.09.2018.

Ime i prezime studenta:	Kristina Javorek
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 845 R, 19.09.2017.
Ephorus podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Analiza primjenjivosti alata za ispitivanje sigurnosti programske podrške**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1 UVOD

Tehnologije se razvijaju velikom brzinom, poboljšavaju se aplikacije i njihove performanse, korisnici imaju sve više izbora i mogućnosti, ali povećanjem aplikacija povećava se i broj sigurnosnih propusta. Gotovo svaka aplikacija ima potencijalne ranjivosti i sigurnosne rizike, a sve veća kompleksnost programske podrške kao i njena dinamičnost otežavaju testerima otkrivanje grešaka te sigurnosni rizici ostaju skriveni. Kako bi došli do podataka, hakeri pokušavaju pronaći te sigurnosne propuste, a kao rezultat toga mnoge web stranice su *hakirane*.

Zbog provala na web stranice i krađe osobnih podataka pojavila se potreba za sigurnijim web stranicama. Testiranje i skeniranje aplikacija je od velike važnosti za osobe koje razvijaju i posjeduju aplikacije kao i za korisnike aplikacija. Korisnici žele osigurati privatnost svojih podataka i biti uvjereni da su oni na sigurnom, a osobe koje razvijaju web stranice žele biti sigurne da su njihove web stranice pouzdane. Zbog tih razloga prije same objave aplikacije potrebno je provesti dobru provjeru i testirati aplikaciju automatski i ručno kako bi se otklonile ranjivosti, poboljšala kvaliteta i procijenila pouzdanost aplikacije. Pri automatskom testiranju web aplikacija, testeri se mogu koristit s nekim od već postojećih sigurnosnih alata za skeniranje koji im nude razne mogućnosti i pomažu otkriti sigurnosne propuste. Postoje mnoge aplikacije čija je glavna svrha upravo provođenje testova i identificiranje potencijalnih sigurnosnih propusta. Aplikacije pomoću kojih se izvodi sigurnosno testiranje se baziraju na principu crne kutije (engl. black box), što znači da ne pristupaju izvornim kodovima već izvode funkcijska testiranja. Uz mnoge verzije koje se naplaćuju, postoje i brojne besplatne sigurnosne aplikacije otvorenog koda. Kada se govori o aplikacijama otvorenog koda misli se na one aplikacije koje programeri mogu modificirati te pomoći u njihovom daljnjem razvoju kako bi se što bolje zadovoljile potrebe korisnika.

U drugom poglavlju razmotreni su najčešći sigurnosni propusti koji se pojavljuju u web aplikacijama, dan je opći pregleda ranjivosti koje hakeri mogu iskoristit za upad u sustav i preuzimanje informacija, primjeri sigurnosnih propusta i savjeti za smanjivanjem tih rizika. Testeri aplikacija pokušavaju raznim testiranjima pronaći potencijalne ranjivosti, a u tom poslu im uvelike može pomoći korištenje sigurnosnih alata. Zbog toga je u trećem poglavlju detaljno analizirano pet sigurnosnih skenera, njihovi osnovni dijelovi, način rada, mogućnosti koje pružaju i sve osnovno što treba znati prije samog procesa skeniranja. Nakon proučavanja skenera može se početi sa skeniranjem web aplikacija. U četvrtom poglavlju, dan je primjer skeniranja web aplikacije korištenjem analiziranih i opisanih skenera. Prikazan je način rada, rukovanje alatima, potrebna podešavanja kao i rezultati samih skeniranja.

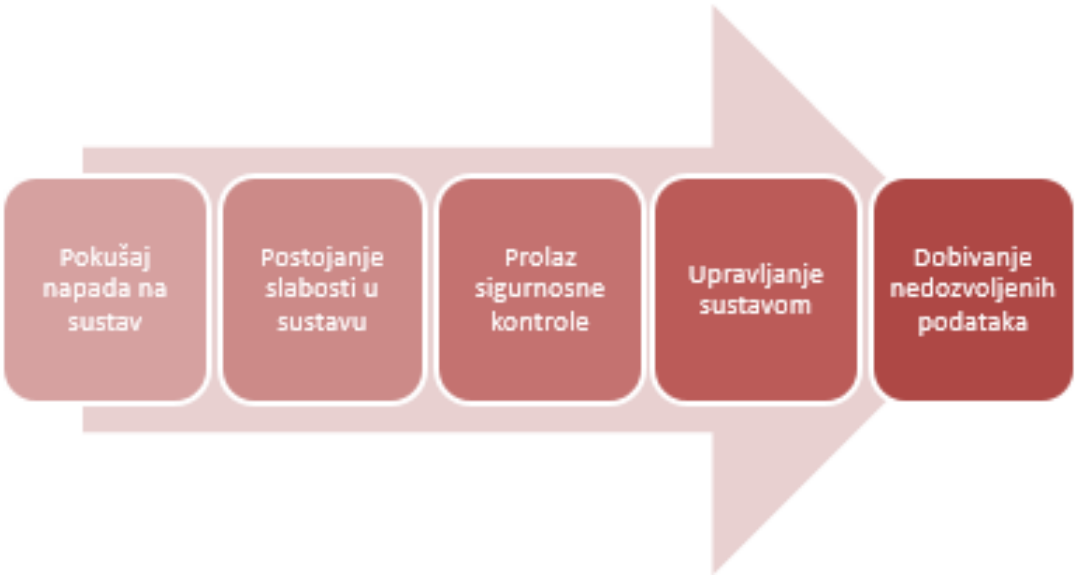
1.1 Zadatak diplomskog rada

U diplomskom radu potrebno je istražiti i analizirati raspoložive alate za ispitivanje sigurnosti programske podrške, kao i tehnike detektiranja i oblike ranjivosti programskog koda. Nadalje, prikladne alate potrebno je implementirati kao klijentske alate ili usluge te ih primijeniti na odgovarajućem programskom projektu web aplikacije i dati procjenu ranjivosti s preporukama za poboljšanje programskog koda.

2 RANJIVOSTI PROGRAMSKOG KODA

Napadači mogu pokušati izvršiti napad na mnogo različitih načina kako bi upravljali sustavom i došli do različitih podataka. Napadi mogu predstavljati veće ili manje rizike za aplikaciju i korisnike. Kako bi se spriječio napad potrebno je na vrijeme otkriti ranjivosti aplikacije i popraviti potencijalne opasnosti. Ponekad je jednostavno otkriti i popraviti moguće putanje napada, a ponekad jako teško. Rezultat napada u nekim slučajevima može biti jako opasan i ozbiljan, dok u nekim drugim slučajevima posljedice napada ne moraju biti značajne.

Na slici 2.1. prikazani su koraci uspješnog napada. Svaki napad započinje pokušajem upada u sustav. Da bi napadač provalio u sustav, mora postojati slaba točka, odnosno ranjivost. Nakon prolaska kontrole, napadač može nedozvoljeno upravljati nekim dijelovima sustava i tako doći do osjetljivih podataka.



Sl. 2.1. *Koraci uspješnog izvršenja napada*

OWASP (engl. Open Web Application Security Project) je organizacija usmjerena poboljšanju sigurnosti softvera. U sljedećim potpoglavljima opisano je 10 najčešćih sigurnosnih rizika web aplikacija prema OWASP-u za 2017. godinu.

2.1 Ubrizgavanje (engl. injection)

Pogreške pri ubrizgavanju (npr. SQL i LDAP ubrizgavanja) se pojavljuju kada se korisnički podaci šalju prevoditelju kao dio naredbe ili upita [1]. SQL (engl. Structured Query Language)

se koristi za pohranu i manipuliranje bazom podataka. LDAP (engl. Lightweight Directory Access Protocol) je aplikacijski protokol koji se koristi za dodavanje i mijenjanje imenika (datoteke ili skupine podataka o korisnicima, datotekama i dr.) preko IP mreže [2]. Napadač može iskoristiti ranjivost polja za unos podataka ili URL-ova koji vrše interakciju s bazom podataka te mijenjati pozadinske SQL naredbe. Podaci napadača mogu prevariti prevoditelja i uzrokovati izvršavanje neželjenih naredbi ili pristup podacima bez odgovarajuće autorizacije te na taj način manipulirati podacima korisnika [3].

Primjeri napada ubrizgavanjem su:

- SQL ubrizgavanje na stranici za prijavu korisnika

Napadač unosi postojeće korisničko ime, a na mjesto lozinke unosi znak ‘ ili -- kako bi manipulirao SQL naredbom. Prijava nije prihvaćena, ali se može pojaviti detaljna lista o pogrešci u kojoj se mogu vidjeti informacije koje se mogu koristiti za izvršavanje napada. Napadač može za lozinku unijeti ‘OR 5=5, gdje je navodnik znak koji ima posebno značenje u SQL naredbi, a uvjet nakon OR operatora je uvijek istinit te rezultat izvođenja može biti prihvaćena prijava korisnika u sustav odakle napadač može doći do daljnjih informacija [3].

- LDAP ubrizgavanje filtera za pretraživanje

Napadač unosi posebne znakove za manipulaciju kao što su: '*', '|', ',', '&' u LDAP filter za pretraživanje. Ako na primjer napadač unese * umjesto korisničkog imena filter će pretraživati i vratiti svaki objekt u kojem se pojavljuje traženi atribut pretraživanja (u ovom slučaju - korisničko ime) bez obzira koju vrijednost sadrži. Na taj način napadaču se mogu prikazati neki ili svi atributi korisnika [4].

Prema [1] i [3], moguće štete napadača su:

- Neovlašteni pristup
- Izvršavanje administracijskih operacija nad bazom podataka
- Brisanje, dodavanje, mijenjanje podataka u bazi podataka
- Čitanje osjetljivih podataka iz baze podataka (korisničko ime, lozinka)
- Krađa podataka
- Unos zlonamjernog sadržaja
- Dodavanje ili izmjena objekata u LDAP strukturi

Preporuke za sprječavanje napada su:

- Pokretanje aplikacije s minimalnim privilegijama
- Ne prikazivati detaljne poruke o pogrešci
- Koristiti mehanizam za validaciju unesenih podataka
- Izbjegavati posebne znakove koje koriste prevoditelji
- Sigurnosno testiranje aplikacije
- Parametrizirani upiti
- Simbole i interpunkcijske znakove koristiti u obliku HTML ekvivalenata

2.2 Neispravna autentifikacija i upravljanje sjednicama (engl. broken authentication)

Pogrešno implementirane funkcije zadužene za autentifikaciju i upravljanje sjednicama mogu dozvoliti napadačima preuzimanje identiteta drugih korisnika ili ugroziti korisničke podatke. Prvi pristup aplikaciji zahtjeva registraciju korisnika, a nakon toga korisnik pristupa aplikaciji unoseći korisničko ime i lozinku. Nakon uspješne prijave u sustav web stranica obično za svaku sjednicu stvara ID i kolačiće sjednice koji sadrže osjetljive podatke (kao što su korisničko ime i lozinka). ID sjednice najčešće podrazumijeva niz (engl. string) nasumičnih slova i brojeva. Web aplikacije trebaju dobro čuvati te podatke kako bi korisnike zaštitile od napadača. Problem predstavljaju sjednice koje se ne poništavaju nakon izlaska korisnika iz sustava zbog čega osjetljivi podaci i dalje postoje u sustavu i dostupni su napadaču. Korisnički podaci su ugroženi ukoliko napadač koristi računalo na kojem postoje neponišteni kolačići zbog toga što aplikacija ne poništava podatke završene sjednice ili se korisnik nije odjavio s računala već je samo zatvorio preglednik. Također problem su i konekcije između korisnika i aplikacije koje nisu enkriptirane zbog čega napadač može presresti informacije koje se prenose između korisnika i web aplikacije bez znanja korisnika. Potrebno je pripaziti i provjeriti lozinku korisnika jer slabe lozinke koje se često pojavljuju omogućuju napadačima lak upad u sustav i preuzimanje identiteta korisnika [3].

Prema [1], primjeri napada neispravne autentifikacije i upravljanja sjednicama su:

- Prijava u sustav bez enkripcije konekcije

Korisnik se prijavljuje u web aplikaciju. Za prijavu je potrebno unijeti korisničko ime i lozinku. Pretpostavimo da veza između korisnika i web aplikacije nije enkriptirana. Nakon uspješne

prijave u sustav spremaju se podaci o korisničkom imenu, lozinki i ID sjednice te su ti podaci vidljivi napadaču. Napadač može te podatke koristiti za prijavu u sustav.

- ID sjednica ostaje važeći nakon izlaska iz sustava

Prilikom prijave u sustav korisnik na javno računalo unosi važeće korisničko ime i lozinku te se nakon nekog vremena odjavljuje. Poslije njega napadač dolazi za računalo. Zbog važeće ID sjednice napadač može pristupiti aplikaciji nakon odjave korisnika jednako kao i kad je korisnik bio prijavljen u sustav bez potrebe za ponovnom prijavom u sustav. U tome slučaju napadač može otvoriti stranicu korisnika i doći do njegovih podataka.

- Prikaz ID sjednice u URL-u

Korisnik se prijavljuje na stranicu za online kupovinu. ID sjednice se nakon uspješne prijave prikazuje u URL-u. Korisnik šalje link nekom drugom korisniku. Kada korisnik kojem je poslan link posjeti dobivenu stranicu, prikazuju mu se podaci pošiljatelja jer aplikacija koristi ID sjednice za identificiranje korisnika koji pristupa aplikaciji.

- Korisnik koristi jednostavnu lozinku

Prilikom prijave u sustav korisniku je omogućeno postavljanje bilo koje lozinke, a sustav ne vrši provjeru jednostavnosti lozinke. Korisnik za lozinku i korisničko ime postavlja administrator što napadaču omogućuje jednostavan upad u sustav. Napadač pokušava pogoditi korisničko ime i lozinku korisnika te se prijaviti u sustav korištenjem jednostavnih, već poznati lozinki. Nakon nekog vremena napadač uspijeva i pristupa aplikaciji koristeći identitet korisnika.

Moguće štete napadača prema [3] su:

- Neovlašteni pristup sustavu
- Otkrivanje i izmjena neautoriziranih informacija
- Presretanje korisničkih podataka
- Krađa podataka
- Preuzimanje identiteta korisnika

Prema [3] preporuke za sprječavanje napada su:

- Enkripcija osjetljivih podataka
- Provjeriti jakost lozinke prilikom postavljanja

- Ne dozvoljavati postavljanje čestih i jednostavnih lozinki
- Omogućiti pravilan oporavak računa i registraciju koji su otporni na napade
- ID sjednice postaje nevažeći nakon odjave korisnika iz sustava
- ID je pravilno pohranjen i ne pojavljuje se u URL-u
- Slanje osjetljivih podataka web aplikaciji korištenjem POST zahtjeva
- Ograničiti broj neuspjelih prijava u sustav

2.3 Izlaganje osjetljivih podataka (engl. sensitive data exposure)

Mnogi osjetljivi podaci (npr. broj kreditne kartice, korisničko ime, lozinka) nisu dobro zaštićeni što ostavlja prostora napadačima za pristup, krađu ili mijenjanje podataka. Slabo zaštićene podatke napadači mogu iskoristiti za npr. prijevare s kreditnim karticama, krađu identiteta ili neke druge zločine. Jedan od načina dolaska do podataka je presretanje podataka koji se šalju preko interneta. Podacima koji su loše enkriptirani ili uopće nisu enkriptirani lako je pristupiti. Također postoji mogućnost pristupa osjetljivim podacima preko drugih ranjivosti programa.

Primjeri napada prema [1] su:

- Presretanje podataka

Korisnik pristupa web stranici zračne luke koju prisluškuje napadač. HTTPS javlja upozorenje korisniku, ali korisnik odlučuje zaobići upozorenje i pristupiti aplikaciji. Prilikom pristupa stranici od korisnika se zahtjeva unos korisničkog imena i lozinke. Nakon unosa korisnik je preusmjeren na stranicu za kupovinu karte preko interneta. Korisnik unosi podatke o kreditnoj kartici te kupuje kartu za let. Informacije o korisniku spremljene su u kolačić. Budući da je komunikacija prisluškivana, a podaci loše enkriptirani napadač može doći do korisničkog imena, lozinke i drugih podataka kao što je na primjer broj kreditne kartice.

- Pristup korisničkim podacima preko druge ranjivosti programa

Korisnik se prijavljuje na stranicu za kupnju proizvoda preko interneta. Prilikom prijave unosi broj kreditne kartice koji se enkriptira automatskom enkripcijom i pohranjuje u bazu podataka. Prilikom učitavanja podataka iz baze oni se automatski dešifriraju. Napadač koristi SQL ubrizgavanje za pristup podacima iz baze. Zbog loše enkripcije napadač dobiva podatke o korisnicima u jasnom obliku.

Prema [3], moguće štete napadača pri izlaganju osjetljivih podataka su:

- Neovlašteni pristup korisničkim podacima
- Preuzimanje korisničkih podataka
- Novčani gubitak
- Krađa identiteta
- Loša reputacija

Preporuke za sprječavanje napada su prema [1]:

- Enkripcija podataka na prijenosnom i aplikacijskom sloju
- Enkripcija podataka u bazi podataka
- Enkripcija podataka koji miruju i koji se prenose
- Koristiti najnovije algoritme za enkripciju
- Korištenje HTTPS-a za prijenos podataka
- Onemogućiti automatsko dovršavanje unosa formi koje pohranjuju podatke
- Ne pohranjivati nepotrebno osjetljive podatke

2.4 Aplikacije koje parsiraju XML dokumente (engl. XML external entity, krat. XXE)

XML (engl. eXtensible Markup Language) se kao format podataka često koristi prilikom pohrane i prijenosa podataka. Nakon učitavanja, slijedi parsiranje XML dokumenta te se prikazuje rezultat. Kako bi se validirao XML dokument koristi se XSD (engl. XML Schema Definition) ili DTD (engl. Document Type Definition) koji se specificiraju u XML dokumentu [5]. Ukoliko postoji mogućnost učitavanja vlastitog XML dokumenta ili uključivanja zaraženih sadržaja u XML dokument, napadač može iskoristiti slabosti DTD-a i XML parsera za poziv svoje vanjske poveznice te dolazak do podataka ili za neki drugi oblik napada [3]. DTD definira strukturu, elemente i atribute XML dokumenta. Unutarnja DTD deklaracija podrazumijeva deklaraciju DTD-a unutar XML datoteke, odnosno unutar <!DOCTYPE> definicije, a ako se radi o vanjskoj DTD deklaraciji tada se unutar <!DOCTYPE> definicije XML-a navodi referenca na DTD datoteku [6]. Referenca na vanjski entitet predstavlja rizik zbog mogućeg napada prilikom prasiranja XML dokumenta. Vanjski entitet može pristupiti lokalnim i udaljenim datotekama preko identifikatora sustava. Prilikom parsiranja entitet koji definira prečicu do određene

vrijednosti se mijenja u odgovarajuću vrijednost. Ako prilikom parsiranja identifikator sustava sadrži zaražene podatke, može doći do otkrivanja povjerljivih informacija [3].

Primjeri XXE napada su:

- DoS (engl. Denial of Service) napad prema [7]:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xxe [
   <!ELEMENT xxe ANY >
   <!ENTITY a1 SYSTEM "file:///dev/random" >
   <!ENTITY a2 "&a1;">
   <!ENTITY a3 "&a2;&a2;">
]>
<xxe>
   &a3;
</xxe>
```

Kad god se pojavi &a3; XML parser će na odgovarajući način pokušati zamijeniti taj entitet. Budući da neki XML parseri ograničavaju količinu memorije koju će koristit ukoliko napadač iskoristi ranjivosti XML entiteta pozivajući entitete unutar entiteta može doći do DoS napada.

- Dobivanje podataka s poslužitelja prema [8]:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xxe [
   <!ELEMENT xxe ANY >
   <!ENTITY attack SYSTEM "file:///etc/shadow" >
]>
<xxe>
   &attack;
</xxe>
```

Napadač može iskoristiti nedostatke XML parsera koji parsira vanjske entitete i postaviti zahtjev za dobivanjem podataka o sustavu. U prethodnom primjeru napadač pristupa datoteci u kojoj se nalaze šifrirane korisničke lozinke. Korištenje nekog drugog entiteta može napadaču omogućiti saznanje o podacima korisnika, izvornom kodu ili uvid u neke druge podatke.

Prema [7], moguće štete XXE napada su:

- Pristup lokalnim i udaljenim resursima

- Pristup podacima u sustavu
- Otkrivanje povjerljivih podataka
- DoS napad
- Krivotvorina zahtjeva poslužitelja
- Pokretanje proizvoljnog koda
- Izvođenje SSRF (engl. Server-Side Request Forgery)

Preporuke za sprječavanje XXE napada prema [8] su:

- Onemogućiti DTD uključen u XML dokument
- Koristiti lokalni statički DTD
- Koristit JSON format podataka
- Validirati postavljeni XML dokument
- Nadograditi XML parsere i biblioteke

2.5 Loša kontrola pristupa (engl. broken access control)

Kontrola pristupa podrazumijeva ograničenja kojima se definira koji korisnici mogu pristupiti kojim podacima. Nedostatak automatizirane detekcije, nepravilno konfiguriranje, nedostatak dobrog funkcionalnog testiranja i loše postavljena ograničenja uzrokuju ranjivost u kontroli pristupa. Kontrolom bi se trebalo osigurati da samo autorizirane osobe mogu izvoditi određene radnje, mijenjati bazu podataka ili dobiti uvid u osjetljive podatke. Nedostatke u kontroli pristupa napadači će pokušati iskoristiti kako bi zaobišli provjeru prilikom pristupa, povisili svoje pravo pristupa i privilegije ili napravili neki drugi nedopušteni napad. Ukoliko se radi o pogrešnoj CORS (engl. Cross-Origin Resource Sharing) konfiguraciji, napadači imaju priliku neovlaštenog pristupa API-u. Najbolji način za detekciju nedjelotvornih kontrola pristupa je ručno, odnosno penetracijsko testiranje [3]. Penetracijsko testiranje predstavlja ovlašteno testiranje, imitaciju napada, tj. pokušaj upada u sustav kako bi se otkrile moguće prijetnje sigurnosti.

Primjeri napada kod loše kontrole pristupa su:

- Promjena datoteke koja se prikazuje [9]

http://attack.com/info/getData?file=page-info.php

http://attack.com/info/getData?file=passwords.php

Napadač mijenja vrijednost parametra kako bi dobio podatke o korisničkim računima. Jednostavnom promjenom datoteke koja će se prikazati zbog neispravne i loše provjere autorizacije, nakon provedbe upita napadač dobiva osjetljive podatke o korisnicima.

- Promjena korisničkih privilegija [3]

http://attack.com/overview

http://attack.com/overview/admin

Napadač mijenja URL stranice u pregledniku i bez prava administratora pristupa administratorskim podacima.

Moguće štete napadača pri lošoj kontroli pristupa su:

- Pristup sustavu kao korisnik ili administrator
- Pristup podacima drugih korisnika
- Pristup osjetljivim podacima
- Kreiranje, brisanje i izmjena podataka
- Korištenje privilegiranih funkcija

Preporuke za sprječavanje napada prema [3] su:

- Penetracijsko testiranje
- Minimizirati korištenje CORS-a
- Implementirati i koristiti mehanizme za kontrolu pristupa
- Provjera autorizacije za svaku stranicu kojoj korisnik pristupa
- Obratiti pozornost na moguće promjene URL-a

2.6 Pogrešna konfiguracija sigurnosti (engl. security misconfiguration)

Napadači mogu pokušati iskoristiti nepravilnu implementacija kontrola koje su zadužene za sigurnost podataka aplikacije kao što su poruke o pogreškama koje sadrže osjetljive podatke, zastarjele verzije sustava, programskog okvira i komponenti, nezaštićene datoteke i direktoriji kako bi dobili informacije i neovlašteno pristupili sustavu. Propusti mogu uzrokovati ugrožavanje cijelog sustava. Kako bi se napadačima onemogućio neovlašten pristup osjetljivim podacima u sustavu ili određenim funkcionalnostima bitno je da sigurnosna konfiguracija bude

ispravno konfigurirana za aplikacije, aplikacijske poslužitelje, platforme, web poslužitelje, programske okvire, baze podataka [3].

Primjeri napada, prema [3], su sljedeći:

- Dobivanje informacija o sustavu

Konfiguracija aplikacijskog poslužitelja omogućuje korisnicima da prilikom pogreške dobiju detaljnu informaciju o pogrešci iz koje se mogu vidjeti potencijalni nedostatci. Napadač iskorištava tako dobivene informacije za upad u sustav.

- Izlistanje direktorija

Na poslužitelju nije onemogućeno izlistanje direktorija. Napadač otkriva i pristupa sadržaju direktorija. Izlistava direktorije kako bi pronašao željenu datoteku te dobio podatke o brojevima kreditnih kartica i sl.

Moguće štete napadača pri pogrešnoj konfiguraciji sigurnosti su:

- Neautorizirani pristup korisničkom računu
- Neautorizirani pristup datotekama
- Dobivanje podataka iz baze podataka
- Pregled osjetljivih podataka
- Dobivanje informacija o aplikaciji za pokušaj drugih napada

Preporuke za sprječavanje navedenog napada prema [1] su:

- Osigurati sigurnost i dobru razdvojenost komponenti
- Promijeniti zadano korisničko ime i lozinku
- Onemogućiti izlistanje direktorija
- Implementirati provjere kontrole pristupa
- Ažurirati programsku podršku
- DAST (engl. Dynamic Application Security Testing)

2.7 Izvršavanje zlonamjerne skripte u web pregledniku korisnika (engl. cross-site scripting, krat. XSS)

Može se pojaviti kada aplikacija preuzima nepouzdane podatke, odnosno ako korisnički uneseni podaci koji se šalju web pregledniku nisu prethodno validirani. Budući da web preglednik ne može procijeniti da li je skripta povjerljiva ili ne, ukoliko postoji prostor za dodavanje proizvoljnog sadržaja napadači mogu pokrenuti izvršavanje zlonamjerne skripte na klijentskoj strani, tj. u web pregledniku korisnika i oteti informacije o kolačićima sjednice, preusmjeriti korisnika na zlonamjernu web stranicu ili izvršiti slične radnje koje mogu uzrokovati neželjene posljedice. Postoje tri oblika XSS napada: preusmjeravajući XSS (engl. reflected), DOM XSS (engl. Document Object Model based) i pohranjeni XSS (engl. stored) [3].

Primjeri XSS napada su:

- Preusmjeravanje na poveznicu (preusmjeravajući XSS)

Napadač ubacuje zlonamjerni kod na poveznicu koja se zatim šalje korisniku putem email-a ili mu se prikazuje prilikom pregleda web stranice. Korisnik posjećuje poveznicu koja sadrži ugrađenu skriptu pri čemu se zlonamjerni kod izvršava u web pregledniku korisnika, a napadač dobiva informacije o korisničkim podacima.

- Unos zlonamjernog koda u DOM (DOM XSS)

DOM (engl. Document Object Model) omogućuje dinamički pristup i mijenjanje dokumenta. Napadač iskorištava ranjivost provjere podataka koji se unose na strani klijenta te izmjenjuje DOM određene web stranice čime u stranicu ubacuje svoju zlonamjernu skriptu. Zlonamjerni kod koji je napadač postavio se ne šalje poslužitelju već ostaje unutar DOM-a. Izvršavanjem skripte napadač može doći do podataka koje može iskoristiti za lažnu prijavu u sustav, npr. bankovni račun korisnika.

- Napad prilikom pregleda web aplikacije (pohranjeni XSS)

Napadač mijenja sadržaj web aplikacije na način da dodaje zlonamjerni kod, odnosno postavlja tajnu zlonamjernu skriptu koja se pohranjuje unutar aplikacije. Aplikacija ne provjerava kod koji je postavio napadač te je i on uključen prilikom korisničkog pristupa aplikaciji. Korisnik otvara web aplikaciju i pregledava njen sadržaj što omogućuje izvršavanje skripte napadača.

Prema [1], moguće štete napadača kod XSS napada su:

- Preusmjeravanje korisnika na zlonamjernu web stranicu
- Pokretanje zlonamjernog softvera na korisničkom uređaju
- Krađa sjednice
- Preuzimanje korisničkog računa
- Preuzimanje podataka iz aplikacije
- Mijenjanje sadržaja stranice

Preporuke za sprječavanje XSS napada prema [1] su:

- Provjera valjanosti unosa
- Filtriranje izlaza
- Kodiranje podataka
- Testiranje ubacivanjem bezopasnog JavaScript okvira upozorenja (engl. alert box)
- Korištenje provjerene biblioteke ili programskog okvira

2.8 Nesigurna deserijalizacija (engl. insecure deserialization)

Serijalizacija je pretvaranje objekta u jednostavan format (npr. XML ili JSON) kako bi se mogao slati mrežom, pohraniti na disk ili u bazu podataka. Nakon slanja, odnosno prilikom čitanja podataka potrebno je izvršiti deserijalizaciju, tj. pretvoriti format u prvobitni oblik. Problem se pojavljuje kad se radi o nepouzdanim korisničkim podacima koje kontrolira napadač. Ako aplikacija ili API deserijaliziraju objekt koji je postavio napadač tada oni postaju ranjivi i mogu omogućiti napadačima udaljeno izvršavanje koda, povećanje povlastica ili im dati mogućnost nekog drugog napada [10].

Primjeri napada kod nesigurne deserijalizacije su:

- Python – pickle

Za serijalizaciju i deserijalizaciju objekata u pythonu može se koristiti izvorni modul pickle. Za serijalizaciju objekta se koristi metoda dumps(), a za deserijalizaciju loads(). Napadač postavlja datoteku koja sadrži proizvoljni kod koji će se izvršiti na korisničkom računalu. Budući da je pickle ranjiv na napade izvršavanja koda, nakon deserijalizacije napadačev kod se izvršava čime se može naštetiti korisniku [10].

- PHP forum

U PHP forumu koristi se serijalizacija PHP objekta za pohranu “super” kolačića. U kolačićima se nalaze podatci o ID-u korisnika, njegovoj ulozi, lozinka i dr. Napadač iskorištava ranjivost za unos svojih podataka i mijenja svoju ulogu u serijaliziranom objektu te umjesto korisnika postavlja da je administrator. Nepouzdani korisnički unosi se prihvaćaju i nakon provedbe deserijalizacije napadač dobiva povlastice administratora [3].

Prema [10] moguće štete napadača kod nesigurne deserijalizacije su:

- Povećanje povlastica napadača
- Zaobilaženje prijave
- Napad ubrizgavanjem
- DoS napadi

Preporuke za sprječavanje navedenog napada su prema [3]:

- Ne prihvaćati serijalizirane objekte iz nepouzdanih izvora
- Validirati korisničke unose
- Nadzirati deserijalizaciju
- Pokretati kod koji se deserijalizira u okruženjima sa niskim povlasticama
- Penetracijsko testiranje

2.9 Korištenje komponenti s poznatim ranjivostima (engl. vulnerable components)

Kako bi programiranje učinili jednostavnijim programeri prilikom razvoja aplikacije često u svoju aplikaciju ugrađuju već postojeću komponentu koja je najčešće otvorenog koda. Međutim te komponente mogu sadržavati sigurnosne probleme. Nedovoljno razumijevanje komponenti koje se koriste u aplikaciji, tj. nepoznavanje verzije komponenti koje se koriste, korištenje ranjivog, zastarjelog ili nepodržanog softver može imati ozbiljni utjecaj na aplikaciju. Budući da napadači mogu iskoristiti nesigurnu komponentu za preuzimanje poslužitelja ili krađu osjetljivih podatak, potrebno je dobro obratiti pažnju na komponente koje se koriste. Problem se može pojaviti i nakon popravka ranjivosti komponente jer kao i drugi korisnici i napadači dobivaju

saznanja o ranjivosti te ukoliko se ne preuzme nova verzija mogu naštetiti aplikacijama koje još uvijek koriste ranije verzije programa.

Primjeri napada kod korištenja komponenti s poznatim ranjivostima su:

- CVE-2017-4971: Spring Web Flow, RCE (engl. Remote Code Execution) ranjivost

Spring Web Flow omogućuje implementaciju tokova MVC web aplikacije. Tokovi predstavljaju niz koraka koji pomažu korisnicima u procesu, odnosno rješavanju nekog zadatka [11]. Objekti pogleda mogu biti ranjivi na napade ukoliko zadana konfiguracija nije promijenjena. Navedenu RCE ranjivost napadači mogu iskoristiti za ubacivanje zlonamjernih izraza koji uzrokuju izvršavanje zlonamjernog koda. Konfiguracija se može mijenjati pomoću XML konfiguracijske datoteke. Za sprječavanje navedenog napada potrebno je nadograditi na 2.4.5. ili noviju inačicu [12].

- CVE-2017-5638: Apache Struts 2, RCE (engl. Remote Code Execution) ranjivost

U Apache Struts 2 programskom okviru koji se koristi za razvoj web aplikacija otkrivena je ranjivost koja je omogućavala udaljenom napadaču ubacivanje naredbi operacijskog sustava u web aplikaciju kako bi ukrali povjerljive informacije [13]. Napada je bio omogućen zbog ranjivost u parseru, a izvršavao se preko zaglavalja "Content-Type" koje označava tip informacije koja se šalje. Pri unosu neispravne vrijednosti u "Content-Type" zaglavlje korisniku se prikazuje greška. Navedenu ranjivost napadači iskorištavaju za manipulaciju zaglavlja "Content-Type" i pokretanje zlonamjernog koda na zahvaćenom poslužitelju. Kako bi se korisnik zaštitio od ove vrste napada treba nadograditi sustav na novu verziju [14].

Moguće štete napadača pri korištenju komponenti s poznatim ranjivostima su:

- Upad u sustav
- Izvršavanje zlonamjernog koda
- Krađa osjetljivih podataka
- Izvršenje nekog drugog napada

Preporuke za sprječavanje navedenog napada su prema [3]:

- Analiza softverskih komponenti
- Statička analiza
- Ukloniti nepotrebne komponente

- Ažuriranje komponenti

2.10 Nedovoljno praćenje i bilježenje (engl. insufficient logging and monitoring)

Nadzor i bilježenje događaja koji bi mogli biti kritični za sigurnost tvrtkama daje kontinuirani trag o problemima aplikacije, pokušajima napada i drugim aktivnostima koji su bitni za sigurnost, tj. daje trag o događajima koji su vezani za datoteke i sustav tvrtke. Ukoliko sustav ne nadzire događanja redovito te ne osigurava bilježenje događaja kritičnih za sigurnost kako bi omogućio uvid u trenutno stanje i njihovo praćenje, ostavlja se prostor napadačima koji takvo stanje mogu iskoristiti za postizanje zlonamjernih ciljeva. U tom slučaju zlonamjerne radnje teže je detektirati, smanjena je učinkovitost upravljanja prilikom napada, a napadačima je omogućeno dulje djelovanje s puno manjom vjerojatnosti otkrivanja napada. Vrlo bitno je da se bilježenje vrši pravilno te da zabilježeni podaci budu potpuni i dostupni kako bi se moglo pravovremeno reagirati, identificirati izvor napada te smanjiti napade na sigurnost [15].

Primjeri napada kod nedovoljnog praćenja i bilježenja su prema [3]:

- Napadač pokušava provaliti u sustav. Ako se ne prati stanje i ne bilježe događaji, teško je otkriti potencijalne napade te su napadaču omogućeni ponovni pokušaji upada u sustav čime se povećava vjerojatnost zlonamjernog ishoda.
- Napadač je otkrio ranjivost aplikacije i uspješno provalio u sustav. U aplikaciji se provodi praćenje i bilježenje događa, ali se događaji ne bilježe pravilno i potpuno, tj. izostavljeni su važni detalji. Zbog navedenog nedostatka napadač je prikriven i omogućen mu je nastavak rada.

Prijetnje i moguće štete napadača kod nedovoljnog praćenja i bilježenja su:

- Česti napadi
- Veliki gubitci

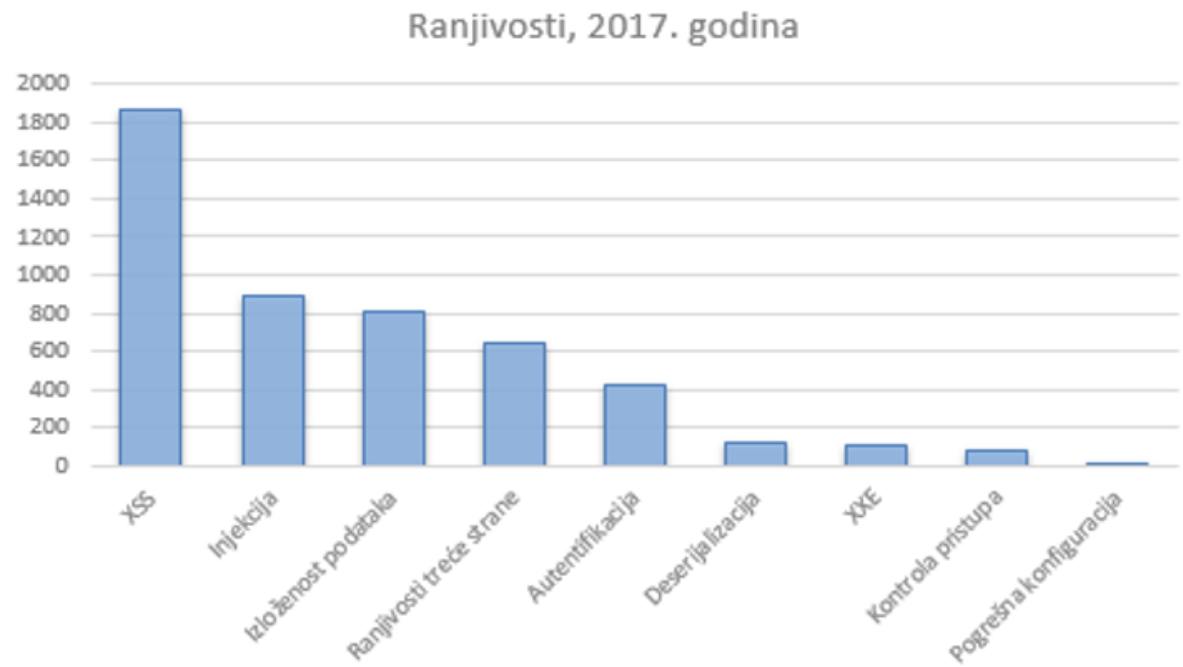
Preporuke za sprječavanje navedenog napada su:

- Provjera zapisa nakon testiranja
- Nadzor trenutnih aktivnosti
- Provoditi pravilno i potpuno bilježenje događaja

- Osigurati dostupnost zapisa
- Uspostaviti plan za odgovor i oporavak

2.11 Pregled i usporedba najčešćih ranjivosti

Prethodno navedene ranjivosti razlikuju se u učestalosti pojavljivanja. Neke se pojavljuju češće od drugih. Na slici 2.2 prikazan je grafikon pojavljivanja najkritičnijih ranjivosti za 2017. godinu.



Sl. 2.2. *Broj pojavljivanja različitih tipova ranjivosti u 2017. godini*

Podatci o učestalosti pojedinih tipova ranjivosti su preuzeti s web stranice navedene u [16].

S obzirom na prikazani grafikon, u 2017. godini se najčešće pojavljivala XSS ranjivost, a najmanje pogrešna konfiguracija. Skoro duplo manje pojavljivanja od XSS napada ima ubrizgavanje iza koje odmah slijedi izloženost podataka. Nešto manje su pojavljivale ranjivost treće strane i napadi kod prijave u sustav. Nesigurna deserijalizacija, XXE i loša kontrola pristupa imaju najmanji broj pojavljivanja odmah iza pogrešne konfiguracije.

U prethodnim potpoglavljima detaljno su opisane najčešće ranjivosti koje je zabilježio OWASP za 2017. godinu. OWASP je usmjeren na otkrivanje najkritičnijih i najozbiljnijih rizika koje prijete web aplikacijama te mogu ugroziti sigurnost organizacija i korisnika. OWASP lista ranjivosti izlazi svakih nekoliko godina, a u tablici 2.1. mogu se vidjeti liste ranjivosti za 2010., 2013. i 2017. godinu.

Tab. 2.1. *Usporedba ranjivosti za 2010., 2013. i 2017. godinu*

	2010	2013	2017
A1	Ubrizgavanje (engl. injection)	Ubrizgavanje (engl. injection)	Ubrizgavanje (engl. injection)
A2	XSS (engl. Cross-Site Scripting)	Neispravna autentifikacija i upravljanje sjednicama (engl. broken authentication and session management)	Neispravna autentifikacija i upravljanje sjednicama (engl. broken authentication)
A3	Neispravna autentifikacija i upravljanje sjednicama (engl. broken authentication and session management)	XSS (engl. Cross-Site Scripting)	Izlaganje osjetljivih podataka (engl. sensitive data exposure)
A4	Nesigurne izravne reference objekata (engl. insecure direct object references)	Nesigurne izravne reference objekata (engl. insecure direct object references)	XXE (engl. XML eXternal Entities)
A5	CSRF (engl. Cross Site Request Forgery)	Pogrešna konfiguracija sigurnosti (engl. security misconfiguration)	Loša kontrola pristupa (engl. broken access control)
A6	Pogrešna konfiguracija sigurnosti (engl. security misconfiguration)	Izlaganje osjetljivih podataka (engl. sensitive data exposure)	Pogrešna konfiguracija sigurnosti (engl. security misconfiguration)
A7	Nesigurno kriptografsko pohranjivanje (engl. insecure cryptographic storage)	Nedostatak kontrole pristupa u funkcijskoj razini (engl. missing function level access control)	XSS (engl. Cross-Site Scripting)
A8	Neuspješno ograničavanje URL pristupa (engl. failure to restrict URL access)	CSRF (engl. Cross Site Request Forgery)	Nesigurna deserijalizacija (engl. insecure deserialization)
A9	Nedovoljna zaštita na transportnoj razini (engl. insufficient transport layer protection)	Korištenje komponenti s poznatim ranjivostima (engl. using known vulnerable components)	Korištenje komponenti s poznatim ranjivostima (engl. using components with known vulnerabilities)
A10	Nepotvrđena preusmjeravanja (engl. unvalidated redirects and forwards)	Nepotvrđena preusmjeravanja (engl. unvalidated redirects and forwards)	Nedovoljno praćenje i bilježenje (engl. insufficient logging & monitoring)

Podatci o ranjivostima dostupni su na OWASP službenim stranicama: [17].

Prema OWASP istraživanjima Neispravna autentifikacija i upravljanje sjednicama, Pogrešna konfiguracija sigurnosti, XSS i Ubrizgavanje su zabilježene za sve tri godine. Izlaganje osjetljivih podataka i Korištenje komponenti s poznatim ranjivostima pojavljuju se u 2017. i 2013. godini, dok se od novih ranjivosti u 2017. godini pojavljuje XXE, Loša kontrola pristupa, Nesigurna deserijalizacija i Nedovoljno praćenje i bilježenje.

Neke navedene nove ranjivosti u sebi uključuju neke ranjivosti prošlih godina, a neke se nove i nisu se pojavljivale prethodnih godina.

3 ALATI ZA ISPITIVANJE SIGURNOSTI PROGRAMSKE PODRŠKE

U ovom poglavlju dan je opis alata koji će se testirati. Prije samog testiranja i skeniranja web aplikacija potrebno je proučiti alate koji stoje na raspolaganju kako bi bilo osigurano da se alati pravilno koriste što vodi do najboljih performansi i naposljetku do pronalaska najvećeg broja ranjivosti. Skeneri web aplikacija su automatizirani alati koji testiraju web aplikaciju u potrazi za potencijalnim sigurnosnim rizicima. Postoje razni dostupni skeneri web aplikacija, od onih koji se plaćaju do onih koji su slobodni za korištenje bez novčane naknade. U ovom poglavlju detaljno su proučeni i opisani sljedeći sigurnosni alati: Arachni, Vega, ZAP, w3af i Nikto. Svi alati su otvorenog koda i mogu se besplatno koristiti. Skeneri web aplikacija su programske podrške koje provode testiranje po principu „crne kutije“ (engl. black box testing) na web aplikacijama kako bi pronašli sigurnosne propuste.

Prikaz principa crne kutije dan je na slici 3.1.

Sl. 3.1. *Princip Crne kutije*

Crna kutija označava testiranje u kojem je kod nepoznanica, program predstavlja kutiju nepoznatog sadržaja s vidljivim ulazima i izlazima. Na ulaz se stavljaju određene ulazne vrijednosti i analiziraju se izlazi. Uspoređuju se očekivane izlazne vrijednosti sa stvarnim izlaznim vrijednostima te se tim načinom vrednuje ispravnost programa. Prednost ove vrste testiranja je da nije ovisna o kodu što znači da kada se dijelovi koda promjene testovi se i dalje mogu koristiti. Neovisnost o kodu omogućuje testovima da se primjenjuju na raznim aplikacijama, a ne da se za svaku aplikaciju moraju pisati novi testovi. Na tom principu rade navedeni skeneri koji se mogu koristiti za testiranje aplikacija neovisno o raznolikosti krajnje uporabe aplikacija. Nedostatak principa crne kutije je propuštanje nekih testiranja i ranjivosti upravo zbog neznanja unutrašnje strukture, odnosno koda aplikacije.

3.1 Alat Arachni

Arachni je Ruby programski okvir za skeniranje i procjenu sigurnosti web aplikacija, otvorenog koda (engl. open source). Usmjeren je na skeniranje web aplikacija kao pomoć pri penetracijskim testiranjima i procjeni sigurnosti web aplikacija. Budući da se radi o alatu s javno dostupnim izvornim kodom, svima je omogućeno i dopušteno dodavati nove, bolje optimizacijske testove te nadogradnja alata. Autor alata je Tasos Laskos [18].

3.1.1 Korisničko sučelje alata Arachni

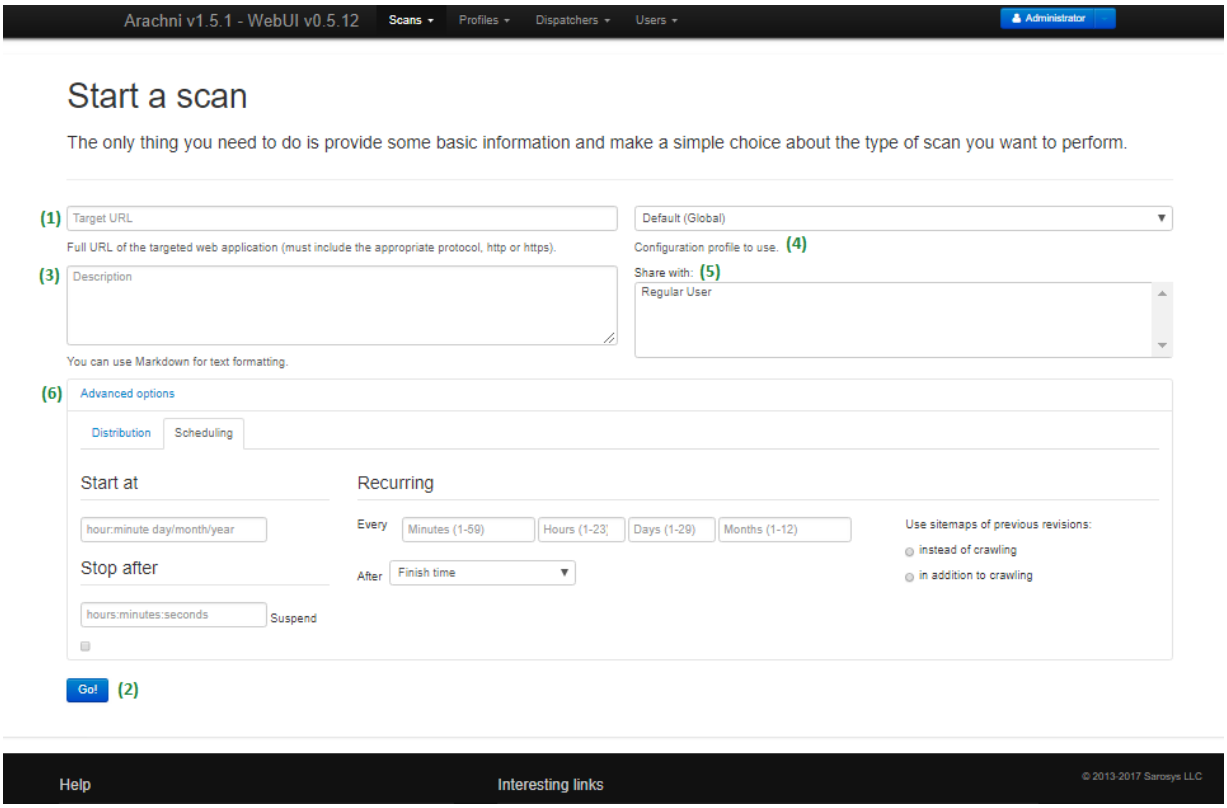
Arachni pruža dva oblika sučelja: preko komandne linije i web korisničko sučelje. Budući da nema ovisnosti kao što su dodatne biblioteke, baza podataka i slično, na vrlo jednostavan način omogućeno je pokretanje i korištenje alata.

Jedna od velikih prednosti alata Arachni je fleksibilnost i visoka modularnost. Budući da je modularan i široke upotrebe, obuhvaća mnogo slučajeva korištenja. Od jednostavnih skeniranja preko komandne linije do velikih skeniranja visokih performansi i višekorisničkih skeniranja web platformi.

Sustav omogućuje višestruke mogućnosti implementacije u rasponu od jednostavnog jednokorisničkog skeniranja preko komandne linije do više korisničkog skeniranja s višestrukim ili paralelnim skeniranjima. Pojednostavljeno, skeniranje se može pokrenuti jednostavnom naredbom u komandnom retku kao što je: $ arachni http://testiraj.webapp.com/ do pokretanja skeniranja preko WebUI [18].

Izgled web sučelja je vrlo jednostavan i lak za razumijevanje. Za pokretanje skeniranja dovoljno je upisati URL aplikacije koju se želi skenirati (1) i kliknuti na gumb „Go!“ (2). Dodatno se mogu postaviti opis (3), konfiguracijski profil (4), dijeljenje skeniranja (5) te napredne postavke (6) koje se odnose na postavljanje vremena ili raspodjelu skeniranja.

Izgled web sučelja prikazan je na slici 3.2.



Sl. 3.2. *Arachni WebUI*

3.1.2 Dodaci omogućeni za alat Arachni

Programski okvir se može proširiti dodavanjem komponenata kao što su: moduli, dodaci, korisnička sučelja i sl. Arachni nije namijenjen samo za sigurnosna skeniranja, već i kao platforma za bilo koju vrstu testiranja „crne kutije“ (engl. black box testing).

Arachni omogućuje dodatke (engl. plug-ins) kao pomoć u automatizaciji zadataka. Ta pomoć odnosi se na široki spektar zadataka, od pomoći pri zadacima prijavljivanja u web aplikaciju do izvođenja visokih meta-analiza. Komponentama sustava kao što su dodaci (engl. plug-ins) omogućuje stvaranje vlastitih tipova komponenti kao i korištenje pogodnosti modularnog dizajna. Preko okvira, dodaci imaju pristup svim Arachni podsustavima i mogu promijeniti ili proširiti ponašanje alata u hodu. Dodaci se pokreću paralelno s okvirom i izvršavaju se neposredno prije početka procesa skeniranja [19].

3.1.3 Učenje preko izvršenih Arachni skeniranja

Vrijeme, preciznost i obuhvat izvještavanja testiranja su poboljšani korištenjem strojnog učenja kojima se poboljšavaju vektori testiranja [20]. Za razliku od većine drugih skenera web aplikacija, Arachni je pametan, tj. trenira se učenjem preko HTTP odgovora koje dobiva. Kako bi ispravno procijenio pouzdanost rezultata i mogao identificirao lažno pozitivne, može

izvršavati meta analize koristeći različite čimbenike. Lažno pozitivni predstavljaju pogrešno prijavljene rezultate u kojima se u rezultatu testiranja pojavljuje prisutnost određene ranjivosti koja u biti nije prisutna.

Arachni uči od ponašanja web aplikacije i prilagođava se svakom resursu web aplikacije, što dovodi do velike pokrivenosti i mogućnosti otkrivanja ranjivosti i malog broja lažno pozitivnih rezultata. Uzima u obzir dinamičku prirodu web aplikacije i može otkriti promjene uzrokovane prijenosom kroz putanje te se u skladu s tim može prilagoditi. Na taj način napad/ulaz vektori koji inače ne bi mogli biti otkriveni od strane alata, već samo od strane ljudi, postaju jednostavno rješivi ovim programskim okvirom [18].

3.1.4 Podržana okruženja alata Arachni

Izvodi asinkrone HTTP zahtjeve, visokih performansi. Alat uključuje integrirano okruženje web preglednika kako bi omogućio dovoljnu pokrivenost za moderne web aplikacije koje koriste tehnologije kao što su HTML5, JavaScript, manipulacije DOM-om, AJAX i dr. Osim nadziranja JavaScript i vanilla DOM okruženja, preglednik podržava popularne programske okvire kao što su AngularJS i JQuery. Time Arachni postaje program za pronalaženje JavaScript i DOM pogrešaka (engl. debugger) s mogućnošću praćenja DOM događaja i JavaScript podataka te samog toka izvršavanja što omogućuje sustavu pokretanje i identificiranje probleme baziranih na DOM-u, kao i praćenje i povezivanje problema s velikim brojem informacija s obzirom na stanje stranice u određenom vremenu.

Trenutno omogućuje rad na sljedećim platformama:

- Operacijski sustavi: MS Windows, Mac OS X i Linux
- Web poslužitelji: Apache, IIS, Nginix, Tomcat, Jetty, Gunicorn
- Programski jezici: PHP, Ruby, ASP, ASPX, Java, Python
- Programski okviri: CakePHP, Rails, Rack, Django, ASP.NET MVC, JSF, Nette, CherryPy i Symfony

Arachni pruža sljedeća sučelja: web korisničko sučelje, korisničko sučelje komandne linije, RPC klijent, RPC poslužitelj, REST poslužitelj [18].

3.1.5 Aktivne i pasivne provjere kod alata Arachni

Arachni obuhvaća preko 40 aktivnih i pasivnih modula koji identificiraju i prijavljuju sigurnosne entitete, tj. prijetnje. Ti entiteti, odnosno prijetnje se kreću od ozbiljnih ranjivosti kao što su

ubrizgavanje i XSS do jednostavnih kao što je pojavljivanje komentara koda na strani klijenta. Aktivni moduli provjeravaju web aplikaciju preko odgovarajućih ulaza, a pasivni moduli traže postojanje datoteka, potpisa i mapa.

Prema [18], trenutno su omogućene sljedeće provjere:

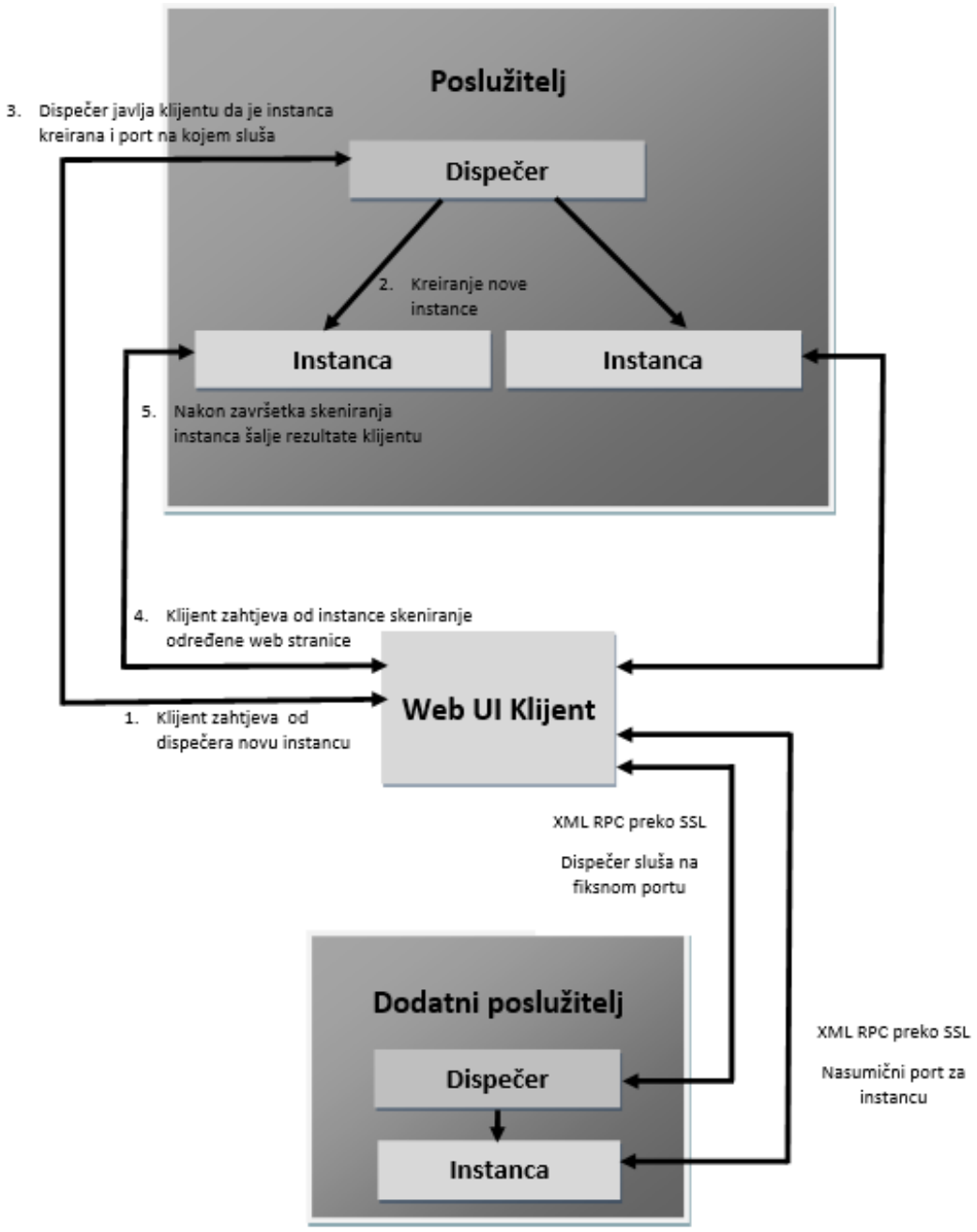
- Aktivne provjere:
    - SQL i NoSQL ubrizgavanja,
    - CSRF detekcija,
    - Ubrizgavanja u kodu (PHP, Ruby, Python, Java, ASP),
    - LDAP ubrizgavanja,
    - Grananje puta (engl. path traversal),
    - Uključivanje datoteka (engl. path inclusion),
    - Razdvajanje odgovora (engl. response splitting),
    - OS naredbena ubrizgavanja,
    - Uključivanje udaljenih datoteka (engl. remote file inclusion),
    - Nedozvoljena preusmjeravanja (engl. unvalidated redirects),
    - XPath ubrizgavanja,
    - Različite XSS provjere,
    - Objavljivanje izvornog koda,
    - XXE.
- Pasivne provjere:
    - Dozvoljene HTTP metode,
    - Sigurnosna kopija datoteka i direktorija,
    - Zajednički direktoriji, datoteke i administracijska sučelja,
    - HTTP PUT i TRACE,
    - WebDAV detekcije,
    - Nedovoljna zaštita transportnog sloja formi za lozinke,
    - Otkrivanje broja kreditnih kartica,
    - Otkrivanje CVS/SVN korisnika (CVS i SVN su dva sustava za kontrolu datoteka koji se koriste u timovima koji surađuju na jednom projektu [21]),
    - Otkrivanje privatnih IP adresa,
    - Stražnja vrata (engl. backdoor),
    - .htaccess LIMIT pogrešne konfiguracije,
    - Neobični odgovori,

- Otkrivanje adresa e-pošte,
- Otkrivanje US ssn (engl. social security number),
- Mješoviti resursi/skriptiranje,
- Izlistavanje direktorija,
- Nesigurni i HttpOnly kolačići,
- Automatsko dovršavanje polja za unos lozinke,
- Prijenos baziran na obrascu,
- Ograničenje u pristupu,
- localstart.asp,
- Kolačići postavljeni za nadređenu domenu,
- Nedostatak zaglavlja,
- Nesigurni CORS,
- Nesigurna pravila o pristupu klijenta,
- Nesigurna politika više domena.

Iako su aktivne i pasivne provjere tehnički identične i dijele isti API, glavna razlika između aktivnih i pasivnih provjera je u tome što aktivne istražuju web stranicu preko vektora kao što su forme, linkovi, kolačići i zaglavlja, dok pasivni moduli istražuju bez potrebe za aktivnom interakcijom s web aplikacijom. Na primjer: SQL ubrizgavanje pripada skupini aktivnih provjera jer je potrebno poslati određeni ulaz web aplikaciji i procijeniti izlaz, dok skeniranje vidljivih adresa e-pošte pripada skupini pasivnih provjera jer se ne vrši interakcija s web aplikacijom, ne šalju se ulazi [19].

3.1.6 Arachni arhitektura

Pojednostavljeni prikaz arhitekture alata Arachni prema [19] i [20] prikazan je na slici 3.3.



Sl. 3.3. *Arhitektura alata Arachni*

Web UI klijent je mozak operacije. On pruža jedinstvenu kontaktnu točku koja kontaktira svoju mrežu dispečerskih poslužitelja (engl. dispatch servers) sa početnim zahtjevima skeniranja. Dispečeri stvaraju novu instancu na poslužitelju. Web UI klijent može direktno komunicirati s kreiranom instancom kako bi je konfigurirao za odgovarajuće poslove skeniranja. Nakon što je instance završila sa zadacima, Web klijent dohvaća i pohranjuje podatke. Resursi koje je instance koristila se vraćaju operacijskom sustavu [20].

Osim web klijenta, dostupna je i opcija testiranja pomoću komandne linije, ali je ta opcija ograničena i nudi nekoliko osnovnih alata za obavljanje skeniranja te najbolje radi pri

implementaciji jednog poslužitelja. Za razliku od skeniranja korištenjem komandne linije, web korisničko sučelje omogućuje višestrukim korisnicima izvršavanje i upravljanje višestrukim skeniranjem te zatim suradnju vezanu za izvršena skeniranja i probleme koji su prijavljeni. Također, olakšano je rukovanje i iskorištavanje alata Arachni te je omogućeno proširivanje opterećenja mnogih skenera preko raznih dispečera [18].

3.1.7 Asinkroni HTTP zahtjevi

Ono po čemu se Arachni razlikuje od drugih sličnih skenera je korištenje asinkronih HTTP zahtjeva u provođenju skeniranja, za razliku od većine skenera koji određeni dio vremena troše na čekanje završetak posla niti za skeniranje. Time je svakoj instance omogućeno paralelno slanje HTTP zahtjeva prema cilju što uvelike poboljšava brzinu te u isto vrijeme može biti pokrenuto više skeniranja. Broj instance koje se može koristiti ovisi o broju dostupnih poslužitelja kao i dostupnosti resursa [20].

3.1.8 Moguće opcije skeniranja u alatu Arachni

Prije samog skeniranja potrebno se ulogirati na web klijent.

Pri tome prema [18] postoje dvije role:

- Administrator

  Račun administratora

  E-pošta: admin@admin.admin

  Lozinka: administrator

- Korisnik

  Račun regularnog korisnika

  E-pošta: user@user.user

  Lozinka: regular_user

Administrator može upravljati svime: korisnicima, konfiguracijom skeniranja (postavljanje globalnih i standardnih profila), skeniranjem, problemima i grupama skeniranja, dispečerima (može postaviti globalne dispečere koji su svima dostupni, te zadane dispečer na razini cijelog sustava), promijeniti postavke (samog skeniranja ili profila). Za razliku od administrator, korisnik ima ograničene privilegije, on može: kreirati, dijeliti i upravljati dispečerima i profilima konfiguracije skeniranja, pokrenuti skenirane, organizirati skeniranja, upravljati, komentirati, dijeliti i spremati izvješća testiranja, diskutirati i pregledavati problem, dobivati obavijesti,

pregledavati aktivnosti te preuzeti, pregledati i komentirati one rezultate skeniranja koje je netko podijelio s njim [18].

Prema [18], opcije skeniranja su:

- Direktno skeniranje – od WebUI stroja na webapp, nije potrebno ništa postavljati,
- Udaljeno – korištenjem dispečera,
- Grid – korištenjem višestrukih dispečera,
- Ponavljanje/Izmjena – ponavlja se završeno skeniranje kako bi se identificirale popravljeni ili novi problem,
- Pregled – kombinira rezultate višestrukih ponavljanja skeniranja za jednostavan pregled i upravljanje.

Skeniranje se ne mora odmah pokrenuti već može biti odgođeno za neko kasnije vrijeme ili predefinirane interval. WebUI je jednostavan, responzivnog dizajna koji odgovara korištenju na računalu, tabletu ili mobitelu.

Arachni je u potpunosti automatizirani sustav. Kada se pokrene skeniranje, odrađuje svoj posao i ne zamara korisnika niti zahtjeva bilo kakvu dodatnu interakciju, a na kraju testiranja rezultate pohranjuje u datoteku koja se kasnije može pregledati u nekoliko različitih formata [18].

3.1.9 Preciznost dobivenih rezultata nakon skeniranja pomoću alata Arachni

Rezultati otkrivanja ranjivosti predstavljaju sposobnost skenera da otkrije različite vrste i oblike ranjivosti, kao i točnost dobivenih rezultata, zbog mogućnosti pojavljivanja lažnih pozitivnih. Lažno pozitivni i lažno negativni rezultati pomažu programeru utvrditi koliko se može osloniti na alat i vjerovati njegovim procjenama.

Arachni ima veliku preciznost pronalaska SQL ubrizgavanja, lokalnog i vanjskog uključivanja datoteka, nedozvoljenog preusmjeravanja, sigurnosnog kopiranja i reflektirajućeg XSS-a. Velika preciznost označava da su prijavljene ranjivosti koje stvarno postoje, odnosno da je alat točno procijenio i prijavio greške. Za navedene ranjivosti alat Arachni ima jako nizak broj lažno negativnih što znači da uglavnom ne prijavljuje ranjivosti koje ne postoje [18].

3.2 Alat Vega

Vega je besplatan alat, otvorenog koda namijenjen za sigurnosno skeniranje i testiranje sigurnosti web aplikacija i pronalazak potencijalnih ranjivosti. Temelji se na grafičkom korisničkom sučelju, napisan je u programskom jeziku Java te se može koristiti na Linux, OS X i Windows operacijskim sustavima. Vega uključuje automatizirani skener za provođenje brzih testiranja i presjedajući proxy za taktičke provjere. Alat se može jednostavno proširiti s modulima koji su napisani u JavaScript programskom jeziku. Razvio ga je Subgraph [22].

Alat također proučava TLS/SSL sigurnosne postavke te pronalazi i predlaže mogućnosti za poboljšanje sigurnosti TLS poslužitelja [22]. TLS i SSL su kriptografski protokoli na aplikacijskom sloju koji omogućuju sigurnu komunikaciju preko Interneta [23].

3.2.1 Značajke alata Vega

Od osnovnih značajki prema [24] potrebno je navesti sljedeće:

- Otvoreni kod
    - EPL 1.0 – Eclipse javna licenca, verzija 1.0
- Jednostavan za korištenje
- Baziran na grafičkom sučelju
    - Dobro dizajnirano grafičko korisničko sučelje,
    - Lako za korištenje
- Višeplatformski alat
    - Linux,
    - Windows,
    - Mac OS
- Proširiv
    - Moduli za detekciju su napisani u JavaScript-u,
    - Korištenjem API-ja lako je stvoriti i uključiti nove module za testiranje ranjivost
- Može se automatski logirati na stranicu kada mu se daju ispravni korisnički podaci
- Baziran na Eclipse RCP-u
    - RCP se odnosi na minimalni skup dodataka potrebnih za izgradnju aplikacije s bogatim klijentom (engl. rich client application) te omogućuje XML konfiguraciju korisničkog sučelja

3.2.2 Način rada koji se primjenjuje kod alata Vega

Vega može raditi na dva načina kao skener ili kao proxy.

- Automatizirani skener

Prvim pokretanjem alat se nalazi u skener modu. Skener je automatizirani alat za skeniranje i testiranje sigurnosti aplikacije. Omogućuje pronalazak poveznica, pretraživanje web stranice i analiziranje sadržaja kako bi pronašao veze i obrasce koje je potrebno testirati. Može pronaći točke ubrizgavanja i pokrenuti JavaScript module kako bi ih analizirao. Vega sadrži ugrađeni alat za indeksiranje te se može prijaviti na web stranicu ukoliko su mu poznati valjani korisnički podaci. Osim automatskog indeksiranja web stranice, pronalazi poveznice, obrađuje obrasce, pokreće module. Mogu testirati XSS ranjivosti, SQL ubrizgavanja i dr. [22].

- Predsjedajući proxy

Kao što i sama riječ označava, predsjeda interakciju između preglednika i aplikacije. Može i ne mora biti omogućen. Ukoliko je omogućen sluša na lokalnom domaćinu (engl. localhost) kao proxy poslužitelj te omogućuje alatu pregled svih zahtjeva i odgovora. Te poruke, odnosno zahtjevi i odgovori se mogu mijenjati. Komunikacija između poslužitelja i klijenta se odvija preko SSL-a. Podešavanje parametara omogućuje aktivno testiranje stranice. Osim toga Vega omogućuje postavljanje prekidnih točaka (engl. breakpoints) kao i kriterija presretanja odlaznih zahtjeva (iz preglednika) i dolaznih odgovora (s poslužitelja) [22]. Struktura klijent-poslužitelj s proxy poslužiteljem prikazana je na slici 3.4.

Sl. 3.4. *Struktura klijent-poslužitelj s proxy poslužiteljem*

Kao što je već spomenuto, proxy poslužitelj ili posrednički poslužitelj je računalo koje predstavlja posrednika između klijenta i glavnog poslužitelja. Proxy može pratiti sve zahtjeve preglednika i sve odgovore poslužitelja. Osim praćenja, alat može presjesti i mijenjati zahtjeve i odgovore prije slanja krajnjoj točki. Kako bi rad s presjedanjem bio omogućen potrebno je prvo konfigurirati preglednik. Kada je preglednik konfiguriran i proxy uključen, može se započeti s njegovom upotrebom [22].

Vega Proxy Skener omogućuje izvođenje aktivnih i pasivnih skeniranja na ciljanim mjestima uočenim tijekom interakcije između klijenta i poslužitelja preko proxy-a. Radi se o djelomično automatskom, korisničkom testiranju sigurnosti aplikacije. Svaka stranica koja odgovara zadanom opsegu te kojoj klijent pristupa preko proxy-a će biti aktivno skenirana. To je korisno pri pregledu zahtjeva koji se šalju putem aktivnog sadržaja (npr. Java ili AJAX) s klijenta na poslužitelj jer automatskim postupkom nije moguće ili je vrlo nezgodno testirati takve zahtjeve. Za korištenje potrebno je konfigurirati proxy, napraviti opseg ciljanja proxy-a i omogućiti proxy skeniranje [22].

3.2.3 Vrste modula kod alata Vega

Alat Vega podržava module za obradu odgovora. Mogu obrađivati odgovore primljene od proxy-a ili skenera, a to se najčešće primjenjuje u potrazi za informacijama. Moduli se mogu omogućiti i za skener i za proxy.

Postoji dvije vrste modula:

- Moduli za obradu odgovora

Moduli za obradu odgovora se pokreću za svaki primljeni HTTP odgovor. Nalaze se u scripts/scanner/modules/response direktoriju. Na sadržaju ili zaglavlju odgovora mogu izvoditi regularne izraze, također mogu pohranjivati informacije u unutarnju bazu i generirati upozorenja. Svaki modul za obradu odgovora mora imati deklariran objekt i run() funkciju. U objektu se treba definirati naziv i tip. Tip objekta modula za obradu odgovora mora biti postavljen na "response-processor". Run funkcija predstavlja ulaznu točku. Prima tri parametra: zahtjev, odgovor i ctx. Zahtjev: objekt zahtjeva koji predstavlja originalni HTTP zahtjev, odgovor: objekt odgovora koji predstavlja originalni HTTP odgovor i ctx: objekt odgovora obrade sadržaja modula, pohranjuje informacije o sadržaju i okolini modula. Uključuju listu pogrešnih konfiguracija sigurnosti, kao što su: izlistavanje direktorija, stranice s pogreškama i dr. [22].

- Osnovni moduli

Osnovni moduli su napisani u programskom jeziku JavaScript. Odnose se na listu modula koji su raspoloživi korisniku prema predefiniranim postavkama, dostupni su kao dio ugrađenog skupa podataka. Izvršavaju testiranja za razne ranjivosti ubrizgavanjem. Svaki osnovni modul mora definirati objekt modula i inicijalizacijsku funkciju. Objekt modula sadrži dva parametra: naziv

i kategoriju. Moduli šalju promijenjene ili potpuno nove zahtjeve alatu za indeksiranje i bilježe povratne pozive za obradu odgovora [25].

Alati za indeksiranje analiziraju stranice, pronalaze i bilježe URI adrese i obrađuju obrasce. Alat Vega dijeli URI strukturu putanje na manje dijelove i analizira ih kako bi otkrio radi li se o direktorijima ili datotekama. Svaki dio putanje se pohranjuje kao čvor zajedno s identificiranom informacijom o tipu u stablastoj strukturi podataka. URI-i koji prihvaćaju POST ili GET parametar se dalje obrađuju i razdvajaju na čvorove polaznih putanja, jedna za svaki parametar. Odabrani osnovni moduli se pokreću jednom za svaki čvor putanje [25].

Povratni pozivi mogu pohraniti informacije u bazu, obavljati analize provjere ranjivosti, generirati upozorenja ili zakazati nove, dodatne zahtjeve. Funkcija povratnog poziva se pokreće kada se vrati odgovor [25].

3.2.4 Ranjivosti koje Vega alat može otkriti

Prema predefiniranim postavkama, Vega skener sadrži module za sljedeća testiranja:

- Ubrizgavanje zaglavlja
- Napadi ubrizgavanja na URL
- Napadi ubrizgavanja na XML
- XSS ubrizgavanja
- SQL ubrizgavanja
- Napadi ubrizgavanja na ljuske (engl. shell)
- Napadi na direktorije
- Uključivanje udaljenih datoteka
- Napadi u obliku niza (engl. string)
- Napadi ubrizgavanja na OS naredbe
- Unutarnje IP adrese
- WSDL detektor
- AJAX detektor
- Identifikator kreditnih kartica
- Automatsko nadopunjavanje formi
- Izloženost osjetljivih podataka
- Otkrivanje informacija
- ...

Za uvid u više modula i ranjivosti koje alta može detektirati potrebno je pokrenuti alat i od ponuđenih odabrati željene module.

3.2.5 Korisničko sučelje alata Vega

Kada korisnik prvi puta pokrene alat, otvara se korisničko sučelje koje sadrži tri glavan prozora:

- Prikaz web stranica (1)

  Nalazi se u gornjem lijevom kutu. U njemu se prikazuje lista web stranica, tj. putova koji su zabilježeni i koji se pretražuju.

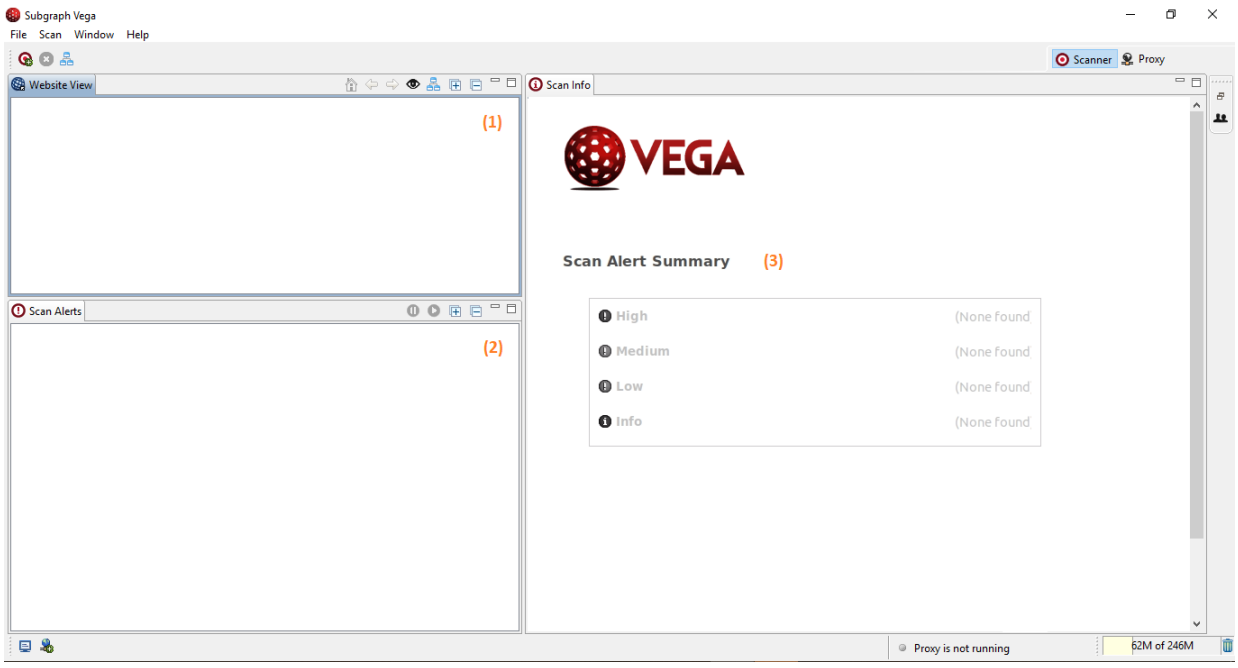
- Upozorenja skeniranja (2)

  Prozor se nalazi u donjem, lijevom kutu. Prikazuje upozorenja svih zabilježenih skeniranja uključujući datum i vrijeme pronalaska upozorenja. Kada se klikne na izlistavanje pojedinog upozorenja skeniranja izlistava se stablo upozorenja te se prikazuju detaljniji podaci o tom skeniranju i zabilježenim potencijalnim ranjivostima.

- Informacije skeniranja (3)

  Glavni prozor u sredini u kojem su potencijalne ranjivosti prikazane prema težini te se također prikazuje i broj otkrivenih ranjivosti. Iznad otkrivenih ranjivosti prikazuje se napredak skeniranja, koliko je skenirano, koliko još treba skenirati i postotak završenog skeniranja.

Slika 3.5 predstavlja prikaz korisničkog sučelja alata Vega:



Sl. 3.5. *Grafičko korisničko sučelje alata Vega*

Prema predefiniranim postavkama alat se na početku nalazi u modu skeniranja. Prebacivanje iz opcije skeniranja na proxy u alatu je vrlo jednostavno. Gumbe za odabir skeniranja ili presjedanja nalazi se s desne strane, iznad glavnog prozora u kojem se prikazuju informacije skeniranja. Klikom na pojedini gumb mijenja se vrsta skeniranja.

Informacije o trenutnim i prošlim skeniranjima pohranjene su i prikazane u radnom prostoru. Ukoliko korisnik odabere čišćenje radnog prostora, uklonit će se svi podaci dobiveni skeniranjima, uključujući upozorenja i sačuvane zahtjeve i odgovore.

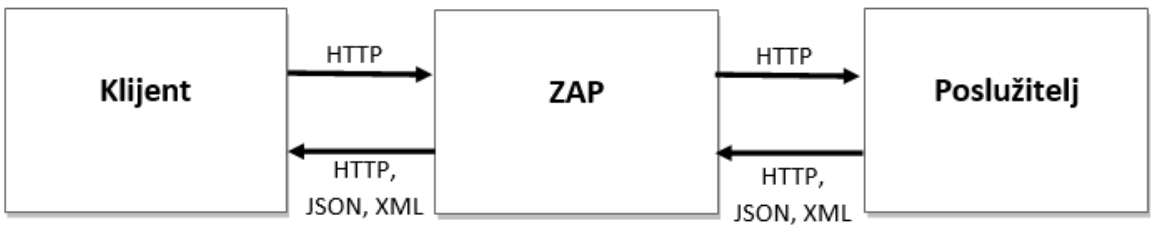
3.3 Alat OWASP ZAP

ZAP je projekt unutar OWASP-a objavljen 2010. godine. OWASP Zed Attack Proxy (ZAP) je besplatan sigurnosni alat za penetracijsko testiranje web aplikacija, otvorenog koda kojeg razvijaju brojni volonteri širom svijeta. Pogodnost ovog alata je mogućnost kombinacije automatskog i ručnog testiranja [26].

3.3.1 ZAP arhitektura

ZAP se koristi kao proxy za presretanje, ispitivanje HTTP poruka, pretraživanje web stranica za otkrivanje URL adresa te provođenje aktivnih i pasivnih skeniranja. Budući da se stalno pojavljuju nove ranjivost, alat ZAP se također konstantno razvija i poboljšava [26].

Slika 3.6 prikazuje ZAP arhitekturu s proxy poslužiteljem:



Sl. 3.6. *ZAP arhitektura*

Alat OWASP ZAP se ponaša kao proxy između klijentske strane, odnosno preglednika i strane poslužitelja. Budući da se nalazi između te dvije strane, može pratiti i bilježiti sve zahtjeve koji prolaze kroz njega prema glavnom poslužitelju, prijenos poruka, promatrati, mijenjati i izvršavati napade. Proxy omogućuje mijenjanje zahtjeva koji su prošli određenu validaciju te se zbog toga

mogu zaobići neka JavaScrit i HTML ograničenja koja aplikacija može sadržavati, a korisnicima omogućuje pregled svih zahtjeva s detaljnim opcijama ispitivanja [20].

ZAP može raditi u prisustvu drugih proxy-a te se može koristiti kao HTTP proxy između web stranice i preglednika korisnika. Kada se ZAP koristi kao proxy između preglednika i poslužitelja, ZAP presijeca HTTP/S promet i omogućuje korisniku izmjenjivanje i preciznije testiranje elemenata aplikacije. ZAP analizira sve poruke, zahtjeve i odgovore, te prijavljuje pronađene, potencijalne ranjivosti.

3.3.2 Automatizirano i ručno testiranje kod alata ZAP

Alat se koristi kao pomoć za automatsko otkrivanje sigurnosnih rizika web aplikacije te osim automatskih skenera pruža i skup alata koji pomažu pri ručnom otkrivanju sigurnosnih propusta.

ZAP može raditi na dva načina:

- Prvi način je automatizirani skener koji se koristi za otkrivanje ranjivosti te se potom generira detaljno izvješće s obzirom na dobivene sigurnosne probleme i rizike. Provođenje standardnog ili brzog skeniranja (engl. quick start) pokreće automatizirano testiranje.
- Drugi način se odnosi na rad ZAP-a kao proxy-a. U tom slučaju ZAP pregledava promet i sve HTTP/S poruke i događaje. Tu se nudi mogućnost izmjene poruka kako bi se mogla analizirati ponašanja koja se razlikuju od predefiniranih.

Kada se alat koristi kao proxy poslužitelj, omogućuje korisniku manipulaciju cijelim prometom koji prolazi kroz njega, uključujući promet korištenjem https-a, a također može se pokrenuti u daemon modu koji se zatim kontrolira preo REST programskog sučelja aplikacije [26].

3.3.3 ZAP Proxy

ZAP proxy se može koristiti za skeniranje, indeksiranje (engl. spider) ili nasumično testiranje (engl. fuzz) aplikacija [20].

Skeniranje – podrazumijeva komunikaciju s web aplikacijom kako bi se razotkrile sigurnosne prijetnje i slabosti arhitekture. Postoje razni oblici skeniranja. Kod skeniranja “crne kutije” (engl. black box test) radi se o testiranju web aplikacije kod kojeg se ne gleda izvorni kod, već se provode stvarni napadi.

Indeksiranje (engl. spider) – odnosi se na posjećivanje i čitanje sadržaja web stranice kako bi se stvorili odgovarajući unosi za pretraživačke alate. Mogu se posjetiti i indeksirati cijele stranice ili određeni dijelovi. Jedan od načina indeksiranja stranice je slijediti sve hipertekstualne veze (engl. hypertext links) na svakoj stranici sve dok nisu posjećene sve stranice [27].

Nasumično testiranje (engl. fuzz) – odnosi se na procese testiranja kod kojih se koristi nasumični ili distribuirani pristup [28]. Koriste se različiti oblici napada kako bi se otkrile sigurnosni propusti. Dovođenje nasumičnih podataka osobito pomaže u otkrivanju propusta i pogrešaka koje se odnose na unos podataka. Također predstavlja dobar alat za testiranje logike aplikacije, mnogih ubrizgavanja, kao i za rukovanje greškama [29].

3.3.4 Preduvjet za korištenje alata OWASP ZAP

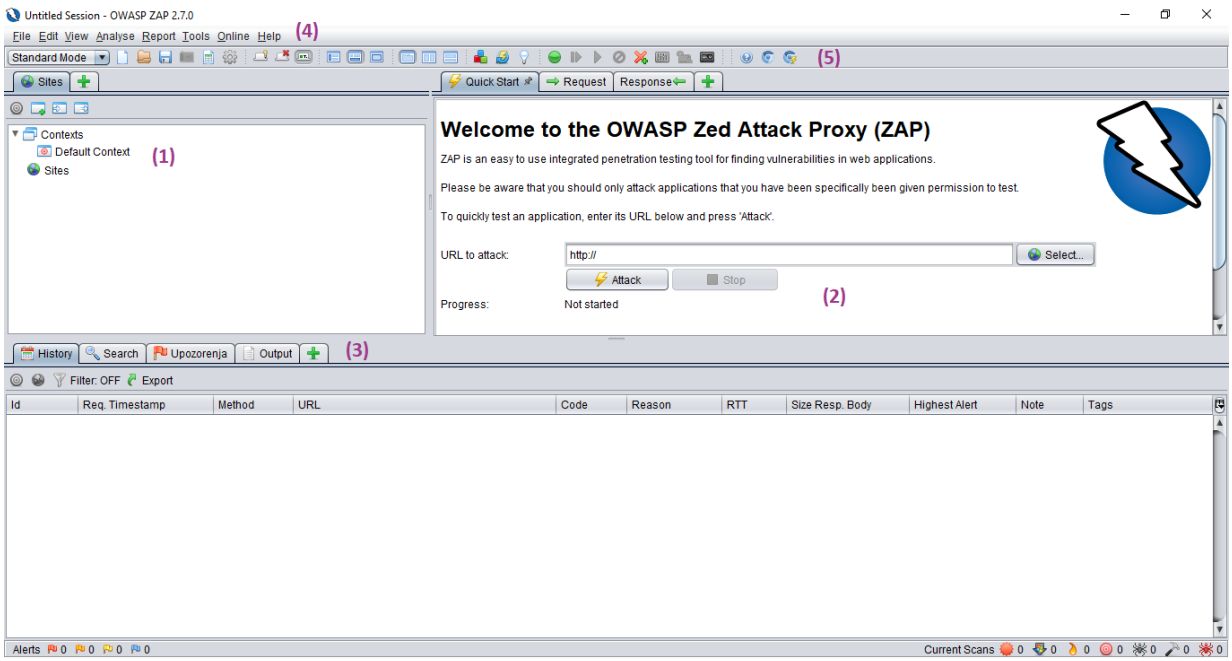
Alat OWASP ZAP je pisan u programskom jeziku Java što ga čini dostupnim za korištenje na mnogim operacijskim sustavima koji podržavaju Javu. Podržan je rad na Linux, Microsoft Windows i Mac OS X operacijskim sustavima. Jedini uvjet za korištenje alata je postojanje Java 7 ili više Java verzije [26].

3.3.5 Sesije alata ZAP

ZAP također omogućuje spremanje sesija kako bi svi snimljeni podaci ostali sačuvani i bili kasnije dostupni. Tijekom rada ZAP pohranjuje podatke u privremenu bazu podataka koja se zatim briše kada se ZAP zaustavi. Ukoliko se želi sačuvati podatke, najbolje je odabrati pohranjivanje sesije prije samog početka testiranja čime se privremena baza podataka kopira u trajnu. Spremljene sesije omogućuju korisniku da napravi pauzu od testiranja te se kasnije vrati i ponovno započne s testiranjima [26].

3.3.6 Korisničko sučelje alata ZAP

Na slici 3.7 prikazano je ZAP korisničko sučelje.



Sl. 3.7. *Korisničko sučelje alata ZAP*

Prema slici 3.7, grafičko korisničko sučelje sadrži tri glavna prozora, a s obzirom na [30] pojašnjenja odjeljaka su:

- Lijevi odjeljak (1) – dio s dva padajuća gumba: "Context" i "Sites". Pod padajućim izbornikom "Sites" pojavljuju se sve web stranice koje se želi skenirati. U "Context" odjeljku se specificira web stranica od interesa.
- Desni odjeljak (2) – ovdje se nalazi URL dio u kojeg je potrebno upisati URL koji se želi testirati. Gumb "Attack" započinje napad na aplikaciju, a gumb "Stop" zaustavlja napad. Gumb "Launch Browser" koji se nalazi iznad URL adrese se koristi ukoliko korisnik želi ručno testirati aplikaciju. U tom slučaju, ZAP korisniku klikom na gumb omogućuje pokretanje preglednika s učitanom URL adresom za ručno testiranje. Svi problem i ranjivosti koji se otkriju se i dalje bilježe, šalju i prikazuju na donjem odjeljku.
- Donji odjeljak (3) – sadrži kartice koje su važne za prikaz aktivnosti tijekom procesa skeniranja ranjivosti. Ispod navedenih kartica nalazi se traka napretka koja prikazuje sami napredak skeniranja, broj zahtjeva koji su poslani te spremanje detalja u CSV format.

  Kartice za prikaz aktivnosti su:

  - Povijest – prikazuje web stranice koje se testiraju,
  - Pretraživanje – omogućuje korisniku pretraživanje prema zadanim parametrima,
  - Upozorenja – detalji o pronađenim ranjivostima,

- Izlaz.

Uz navedena tri glavna prozora, ZAP prikazuje:

- Izbornik na vrhu (4) – omogućuje pristup automatiziranim i ručnim alatima,
- Alatnu traka na vrhu (5) – uključuje tipke za brzi pristup često korištenim značajkama.

3.3.7 Izvršavanje napada pri ZAP testiranju

ZAP se može koristiti za aktivno skeniranje preko "Quick Start" kartice ili se može koristit kao proxy alat za manipuliranje stranicom. Najbrži i najjednostavniji rad sa alatom ZAP je preko kartice Brzi početak (engl. quick start). Korisnik unosi jedan URL stranice koju želi testirati. Nakon klika na gumb Napad (engl. attack) ZAP započinje s napadom na aplikaciju. Za indeksiranje aplikacije koristi svog pauka (engl. spider) koji automatski pasivno skenira sve otkrivene stranice, a nakon toga koristi aktivno skeniranje za napad na sve stranice aplikacije. Ovo je dobar način za izvođenje i dobivanje početne procjene, ali sadrži neke nedostatke kao što je manjak kontrole nad fazama napada i istraživanja [26].

Pauk ili indeksiranje se koristi za istraživanje aplikacije i njenog sadržaja. Budući da alat može napasti samo one dijelove aplikacije kojih je svjestan, potrebno je na početku istražiti sve dijelove aplikacije. Ukoliko se koristi kartica za brzi početak, biti će uključeno uobičajeno indeksiranje preko HTML-a. Iako je takvo indeksiranje brzo često nije efektivno kod AJAX aplikacija gdje se linkovi generiraju korištenjem JavaScript-a. Za takve aplikacije, koje koriste AJAX, bolje je koristiti sporije AJAX indeksiranje. Pri takvom indeksiranju kreće se od preglednika i slijede se generirane poveznice. Iako je indeksiranje dobro kao pomoć pri istraživanju aplikacije, potrebno ga je koristiti i kombinirati s drugim načinima ispitivanja aplikacije jer se indeksiranjem provjeravaju aplikacije prema zadanim, predefiniranim podacima dok korisnik može unijeti i testirati aplikaciju na relevantnije informacije. Kako bi ručno istraživanje aplikacije bilo omogućeno, potrebno je konfigurirati preglednik na proxy preko alata ZAP te na taj način omogućili korištenje ZAP-a kao proxy-a [26].

Nakon izvršenog skeniranja mogu se generirati HTML, XML, JSON i MD (engl. MarkDown) izvješća.

3.3.8 Lažno pozitivne i lažno negativne prijave alata ZAP

Jedna od brojnih značajki alata ZAP je njegova osjetljivost i agresivna testiranja koja se mogu ručno, prema potrebi, konfigurirati.

Postoje tri razine osjetljivosti:

- visoka,
- srednja,
- niska.

Ako se pojavljuje veliki broj lažno pozitivnih, može se postaviti razina upozorenja na visoku što opet s druge strane može povećati broj lažno negativnih, odnosno može se pojaviti veći broj ne prijavljenih ranjivosti. U drugom slučaju, ako je problem veliki broj lažno negativnih, razina se može postaviti na nisku [31].

3.3.9 Aktivno i pasivno ZAP skeniranje

Aktivnim skeniranjem pokušavaju se pronaći ranjivosti aplikacije na temelju poznatih napada te zato aktivno skeniranje ne bi trebalo vršiti na aplikacijama koje korisnik koji testira aplikacije ne posjeduje ili na kojima ovlašteno ne radi. Budući da se radi o preddefiniranim napadima, aktivnim skeniranjem se mogu pronaći samo određene vrste ranjivosti. Uz aktivno skeniranje, potrebno je vršiti i ručno skeniranje kao bi se osiguralo da su pronađene sve vrste ranjivosti [26].

Pasivno skeniranje ne mijenja poruke (ni zahtjeve ni odgovore) ni na koji način te je stoga sigurno za korištenje. Prema predefiniranim postavkama, svi HTTP zahtjevi i odgovori koji se šalju web aplikaciji se pasivno skeniraju. Ova vrsta skeniranja je dobra za pronalaženje ograničenog broja mogućih ranjivosti aplikacije, kao što je nedostatak HTTP zaglavlja te predstavlja dobar način za dobivanje uvida u sigurnost web aplikacije kao i dijelova aplikacije na koje se treba više usredotočiti te provesti više testiranja. Kako se ne bi usporavalo testiranje aplikacije, skeniranje se odvija u pozadinskoj niti. Omogućeno je automatsko dodavanje i podizanje upozorenja vezano za potencijalne greške. Skup pravila za automatsko označavanje je također omogućen prema početnim postavkama. Ukoliko je potrebno, moguće je promijeniti osnovne predefinirane postavke pasivnog skenera te postavke automatskog označavanja i podizanja upozorenja [26].

3.3.10 Podešavanje ZAP konfiguracije

Najbrži način za početak korištenja ZAP alata je korištenje predinstaliranog dodatka “Quick Start”. Na taj način omogućeno je jednostavno skeniranje korištenjem URL adrese. Za dublje testiranje potrebno je podesiti preglednik ili funkcijske testove kako bi se mogli povezati sa željenom web aplikacijom koju je potrebno testirati. Ukoliko je potrebno, može se podesiti i sam ZAP za spajanje preko nekog drugog proxy-a. Ukoliko je preglednik pravilno konfiguriran,

omogućeno je jednostavno pokretanje preglednika preko “Quick Start” kartice. U suprotnom slučaju, ako je potrebno koristit neki drugi preglednik, potrebno je ručno konfigurirati preglednik na proxy preko ZAP-a. Nakon što je ZAP konfiguriran potrebno je pokrenuti web aplikaciju koju se želi testirati i provjeriti može li se izvršiti spajanje s tom web aplikacijom. Ako se nije moguće spojiti, potrebno je ponovno provjeriti postavke proxy-a u pregledniku kao i ZAP proxy postavke. Kada je aplikacija uspješno pokrenuta i postoji konekcija moguće je započeti sa penetracijskim testiranjem [26].

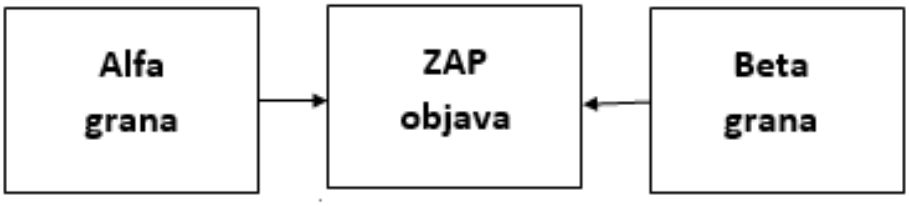
3.3.11 Dodaci u alatu OWASP ZAP

Dodaci, kao što i sama riječ govori, dodaju nove, dodatne funkcije ZAP-u. Kako bi dodatak bio dostupan za korištenje, mora postojati u “plugin” direktoriju koji se nalazi u instalacijskoj ili korisničkoj datoteci. Budući da dodaci imaju pristup unutarnjim dijelovima ZAP-a, mogu pružiti vrlo moćne nove značajke. Ukoliko se želi dodati ili ukloniti neki dodatak, to je moguće vrlo jednostavno učiniti, bez potrebe za ponovnim pokretanjem ZAP alata [26].

Postoje tri oblika statusa dodijeljenog dodacima:

- Alfa – označava da se radi o ranoj fazi razvoja,
- Beta – označava dosta dobru kvalitetu, ali postoji određena nedovršenost,
- Objava (engl. release) – označava da se radi o visokoj kvaliteti i da proizvod služi svojoj svrsi.

Struktura radnog prostora alata ZAP prikazana je na slici 3.8.



Sl. 3.8. *Struktura radnog prostora*

ZAP je podijeljen na alfa i beta granu kako se dodaci, tj. proširenja koja još nisu u potpunosti dovršena ne bi miješala s već dovršenim proširenjima. Na ovaj način omogućeno je jednostavno korištenje proširenja, kao i uvid u trenutno stanje razvoja. Na početku svog razvoja dodatak se nalazi u alfa grani, zatim kada je više razvijen i blizu završetka prelazi u beta fazu, a kada je razvoj dodatka u potpunosti dovršen prelazi u završnu fazu i spreman je za objavljivanje [32].

3.3.12 Otkrivanje ranjivosti alatom OWASP ZAP

Prema [33], ZAP pomaže u otkrivanju sljedeći ranjivosti:

- Ubrizgavanje
    - Automatizirani testovi: SQL ubrizgavanja i aktivno skeniranje
    - Ručni testovi: Fuzzer
- Neispravna autentifikacija i upravljanje sjednicama
    - Automatizirani testovi: testiranje kontrole pristupa
    - Ručni testovi: http sesije, pauk, tj. indeksiranje, nedozvoljeno pregledavanje podataka i generator tokena
- Izlaganje osjetljivih podataka
    - Automatizirani testovi: aktivno i pasivno skeniranje
- XXE
    - Automatizirani testovi: aktivno skeniranje
- Loša kontrola pristupa
    - Automatizirani testovi: aktivno i pasivno skeniranje
    - Ručni testovi: HttpsInfo, skeniranje porta, detekcija tehnologije
- Pogrešna konfiguracija sigurnosti
    - Ručni testovi: spider, ajax spider, kontrola pristupa, HttpsInfo, usporedba sesija
- XSS
    - Automatizirani testovi: aktivno skeniranje
    - Ručni testovi: Fuzzer i Plug-n-Hack
- Nesigurna deserijalizacija
    - Automatizirani testovi: aktivno skeniranje i rukovanje Java serijalizacijom
- Korištenje komponenti s poznatim ranjivostima
    - Automatizirani testovi: pasivno skeniranje
    - Ručni testovi: detekcija tehnologije
- Nedovoljno praćenje i bilježenje
    - Automatizirani i Ručni testovi: spider, aktivni skeneri, fuzzer i dodaci za kontrolu pristupa

Mnogi navedeni dodaci nisu automatski dodani u objavljenu verziju pa ih je potrebno ručno preuzeti i uključiti u testiranje.

3.4 Alat w3af

W3af je skraćenica od engl. Web Application Attack and Audit Framework, a predstavlja podršku za testiranje web aplikacija, pronalazak i prikaz ranjivosti koje mogu sigurnosno ugroziti aplikaciju i korisnike. Kombinira statičke analize koda s testiranjem crne kutije (engl. black box testing).

W3af je napisan u programskom jeziku Python kako bi bio jednostavan za korištenje i proširivanje. Alat je otvorenog koda te se nalazi pod GPLv2.0 licencom [34]. Pomaže u otkrivanju preko 200 ranjivosti: SQL ubrizgavanja, upravljanjem operacijskim sustavom, uključivanju udaljenih datoteka, uključivanju lokalnih datoteka, XSS, prebacivanje nesigurnih datoteka i dr.

3.4.1 Glavne značajke alata w3af

Prema [35], alat w3af može raditi na sljedećim platformama:

- Radi na svim platformama koje podržavaju Python
- Linux, Mac OSX, FreeBSD i OpenBSD
- Kompatibilan s Windowsima

Prednosti w3af alata su:

- Modularnost i fleksibilnost

Nedostatci w3af alata su:

- Visok broj lažno negativnih (engl. false negative)

Za razvojne programere koji žele pridonijeti i proširiti alat preko dodataka, omogućene su sljedeće značajke [34]:

- Daemon – w3af implementira web i proxy poslužitelje koji se jednostavno integriraju u kod
- Brzi HTTP klijent – urllib2 i puno nastavaka omogućuju slanje posebno oblikovanih HTTP zahtjeva velikom brzinom.
- Zapisivanje izlaza – izlaz može biti zapisan u konzolu, kao tex, CSV, HTML ili XML datoteke ili poslan preko e-pošte.

- Fuzzing – alat omogućuje unos korisničkih podataka u razne dijelove HTTP zahtjeva (zaglavlja, kolačići, URL put i naziv datoteke, POST-podatke i dr.).
- Baza znanja – baza znanja se dijeli između dodataka što znači da sve ranjivosti i otkrivene informacije otkrivene jednim dodatkom nakon pohrane u bazu znanja bit će dostupne ostalim dodacima.
- Parsiranje – alat pokušava analizirati, parsirati i izvući veze i obrasce iz bilo kojeg pronađenog HTML-a

3.4.2 W3af arhitektura

W3af programski okvir se može podijeliti u tri glavna dijela, a to su [34]:

- Jezgra - rukovodi radom cijelog procesa i osigurava biblioteke potrebne za dodatke,
- Dodaci – koriste se za pronalazak veza i ranjivosti aplikacije,
- Korisničko sučelje - dozvoljava korisniku podešavanje i pokretanje skeniranja.

Postoje dvije vrste podešavanja alata [34]:

- Globalno podešavanje

Globalnim podešavanjem mijenja se ponašanje jezgre. Postoje dva oblika: http-settings i misc-setings. Sve postavke na početku imaju neku zadanu vrijednost. Ukoliko se postavke nepravilno podese, može doći do smanjenja performansi skenera.

- Podešavanje dodataka

Kao i kod globalnih postavki, postavke dodataka su predefinirane, tj. imaju postavljenu neku zadanu vrijednost. Ukoliko korisnik želi promijeniti predefinirane postavke, trebao bi se dobro informirati i ukoliko je potrebno izvorni kod dodataka kako bi razumio što će se točno dogoditi promjenom.

Nakon promjeni postavki, korisnik može sačuvati svoje promjene preko profila i pokretati testove više puta, a u nekim slučajevima s drugačijim konfiguracijama. Kreiranje, spremanje i učitavanje profila se obavlja preko korisničkog sučelja.

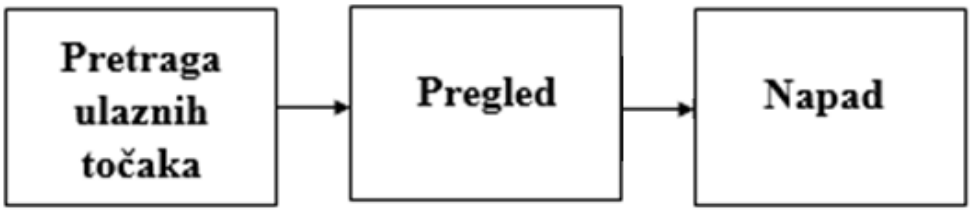
3.4.3 Dodaci u alatu w3af

Dodaci su pisani u Python-u. Proširuju okvir na razne načine kao što su pronalazak novih ranjivosti, otkrivanje novih URL adresa i mnoge druge te šalju posebne HTTP zahtjeve kako bi se otkrile pogreške. Preko baze znanja međusobno dijele informacije.

Neki od dostupnih dodataka su [34]:

- Pregled i provjera
    - LFI (engl. Local File Inclusion),
    - RFI (engl. Remote File Inclusion),
    - SQL ubrizgavanja,
    - CSRF (engl. Cross-Site Request Forgery),
    - Prijenos datoteka,
    - SS ubrizgavanja (engl. Server Side),
    - XPATH, SSS i dr.
- Autentifikacija
    - Opća,
    - Detaljna
- Infrastruktura
    - allowed_methods,
    - detect_reverse_proxy,
    - detect_transparent_proxy,
    - favicon_identification,
    - server_header, server_status i dr.
- Izlaz
    - console,
    - text_file,
    - csv_file,
    - html_file,
    - xml_file,
    - email_report,
    - export_requests
- I mnogi drugi…

Dodaci su podijeljeni u različite skupine, a tri osnovne vrste dodataka su: pretraga ulaznih točaka (engl. crawl), pregled (engl. audit) i napad (engl. attack). Na slici 3.9 prikazane su osnovne vrste dodataka te njihov redoslijed izvršavanja.



Sl. 3.9. *Redoslijed izvođenja osnovnih vrsta dodataka*

Pretraga ulaznih točaka (engl. crawl) – zaduženi su za pronalazak URL-ova, obrazaca i drugih ulaznih točaka. Primjer je pauk (engl. spider) koji uzima korisnički uneseni URL i na temelju njega pretražuje i vraća ulazne točke [35].

Pregled (engl. audit) – na temelju pronađenih ulaznih točaka iz prethodne faze pokušava pronaći pogreške i ranjivosti aplikacije tako što svim pronađenim ulazima šalje posebno oblikovane podatke. Primjer je dodatak za traženje SQL ubrizgavanja [35].

Napad (engl. attack) – Nakon završetka skeniranja korisnici mogu koristiti dodatke za napad na utvrđenim ranjivostima. Koriste ranjivosti pronađene u prethodnom koraku pohranjene u bazu znanja i pokušavaju ih iskoristiti. Ovi dodaci pokreću lokalni proxy koji koristi udaljeni poslužitelj kao izlaznu točku za HTTP zahtjeve. Primjer je sqlmap dodatak koji pretražuje i koristi ranjivosti SQL ubrizgavanja [35].

Od ostalih dodataka ističu se [35]:

Infrastruktura (engl. infrastructure) – identificira informacije o ciljanom sustavu kao što su operacijski sustav i instalirani WAF (engl. Web Application Firewalls).

Dodatak za pretraživanje (engl. grep) – Analizira HTTP zahtjeve i odgovore koje šalju drugi dodaci i pronalazi ranjivosti

Izlaz (engl. output) – pohranjuju informacije u određenom format koji se zatim prikazuje korisniku.

Mangle – omogućuje mijenjanje zahtjeva i odgovora baziranih na osnovnim izrazima.

Uzastopno pokušavanje (engl. bruteforce) – koristi se za probijanje pronađenih zaporki.

Izbjegavanje (engl. evasion) – Mijenja HTTP promet koji stvaraju drugi dodaci i na taj način izbjegava jednostavna pravila za otkrivanje upada.

3.4.4 Korisničko sučelje alata w3af

Korisničko sučelje pruža automatizirane značajke za skeniranje i alate za napredne korisnike koji omogućuju slanje prilagođenih HTTP zahtjeva i druge značajke.

Postoje dva korisnička sučelja, a prema [35] to su:

- Korisničko sučelje konzole

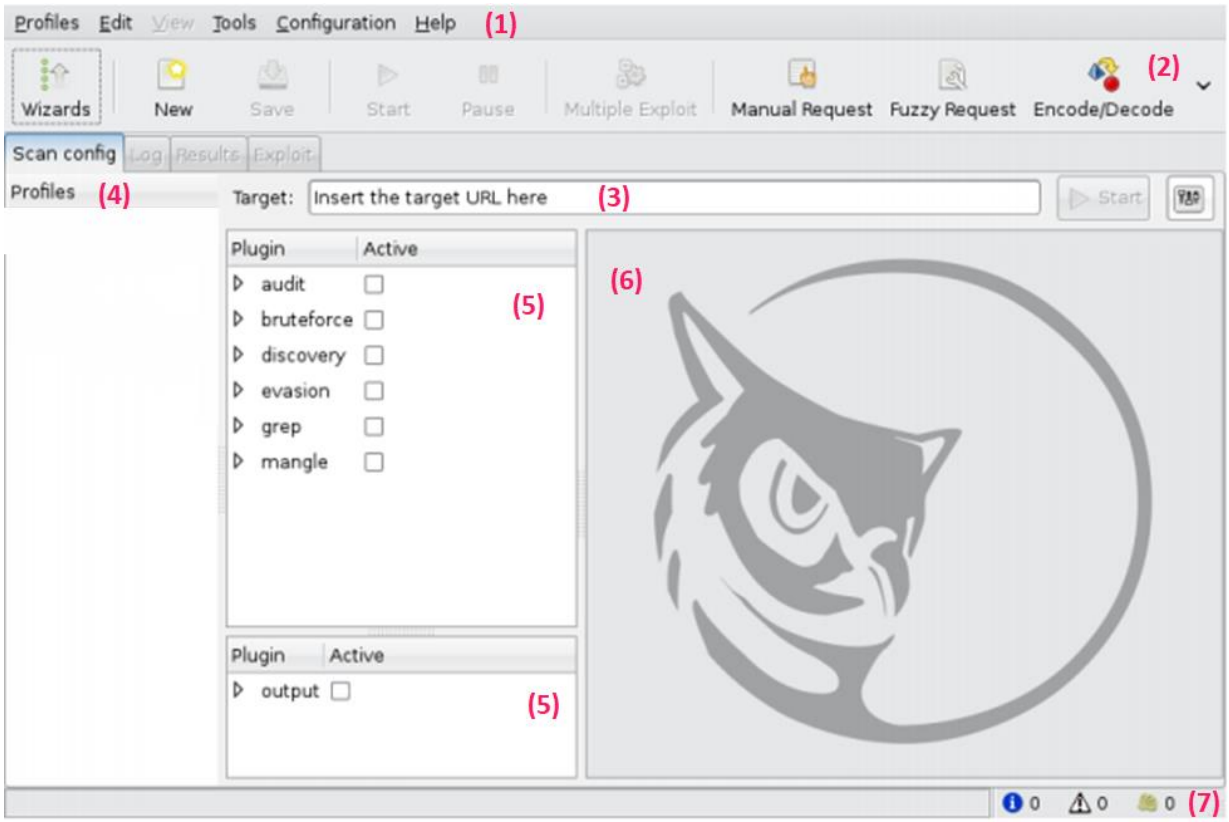
Svi konfiguracijski izbornici omogućuju četiri naredbe:

  - Pregled (engl. view) – prikazuje dostupne opcije i vrijednosti parametara,
  - Postavljanje (engl. set) – postavlja zadanu vrijednost parametra,
  - Pomoć (engl. help) – daje dodatne, detaljne informacije o parametru i
  - Nazad (engl. back) – omogućava prijelaz u prethodni izbornik.

- Grafičko korisničko sučelje

Omogućuje korisniku izvođenje raznih akcija i značajki na brži i jednostavniji način. Omogućuje pokretanje skeniranja i analiziranje rezultata. Kada se sustav učita korisniku su vidljivi: izbornik s mogućnošću namještanja profila, odabira alata, konfiguracijskim postavkama, mogućnošću mijenjanja i traženja pomoći (1), ispod izbornika se nalazi alatna traka s često korištenim značajkama za brži rad (2), ispod alatne trake nalazi se prostor za unos adrese aplikacije koju je potrebno skenirati (3), s lijeve strane nalaze se profili (4) i izbornici dodataka (5), a s desne strane se nalazi prostor za konfiguraciju dodataka (6) te na kraju na dnu prozora se nalaze indikatori pronađenih elemenata (7).

Korisničko sučelje w3af prikazano je na slici 3.10.



Sl. 3.10. *Grafičko korisničko sučelje alata w3af*

3.4.5 Faze w3af skeniranja

Skeniranje se odvija izvršavanjem nekoliko koraka. Dodaci koji se koriste pri skeniranju se pozivaju po točno određenom redoslijedu.

Na početku korisnik unosi URL stranice koju želi skenirati. Pomoću dodatka za indeksiranje, npr. web_spider, alat pokušava pronaći sve URL-ove, parametre niza upita i obrasce u aplikaciji. Nakon mapiranja aplikacije dodaci za pregled i provjeru aplikacije šalju posebno oblikovane nizove svakom parametru. Time se nastoje pronaći greške u aplikaciji koji se zatim prijavljuju korisniku. Na kraju skeniranja korisniku je dostupno izvješće o pronađenim ranjivostima i greškama preko dodatka za izlaz. Korisnik može pregledati izvješće u raznim formatima.

3.4.6 Preduvjeti korištenja alata w3af

Prije same instalacije i korištenja alata potrebno je imati instaliran Git, Python 2.7 i Pip 1.1 [34].

Git se koristi za preuzimanje izvornog koda.

W3af programska podrška se može pokrenuti na svim uređajima koji podržavaju Python. Pokretanje je testirano na raznim Linux varijantama, Mac OSX, FreeBSD i OpenBSD sustavima, a može se instalirati i na Microsoft Windows operacijskom sustavu.

Ažuriranje podrške je jednostavno, a može se izvršavati ručno ili automatski. Kod automatsko ažuriranja u postavkama se može podesiti dnevno, tjedno ili mjesečno automatsko ažuriranje.

3.4.7 Provođenje w3af skeniranja

Kod alata w3af koristi se pristup Crne kutije (engl. black box) za skeniranje web aplikacija. Pojednostavljeno, za provođenje skeniranja potrebno je na početku otkriti sve veze, obrasce i parametre upita. Zatim se svakom polju za unos šalje posebno oblikovani niz i analizira se dobiveni izlaz. Na kraju se izrađuje izvješće koje sadrži rezultate skeniranja.

3.4.8 Otkrivanje ranjivosti pomoću alata w3af

Pruža zaštitu protiv te provjerava:

- SSI (engl. Server-Side Includes)
- SSJS ubrizgavanje (engl. Server Side JavaScript)
- SQL ubrizgavanje
- PXSS (engl. Prevent Cross Site Scripting)
- RXSS (engl. Reflected Cross Site Scripting)
- JSON zaglavlje
- Uključivanje lokalnih datoteka
- Uključivanje udaljenih datoteka
- Preusmjeravanje
- Prijenos datoteke
- LDAP (engl. Lightweight Directory Access Protocol)
- CRLF ubrizgavanje (engl. Carriage Return Line Feed)
- XPATH ubrizgavanje
- Mx ubrizgavanje
- Sjednice
- FORMAT ubrizgavanje
- Sigurnosna pohrana datoteka

3.5 Alat Nikto

Nikto je skener otvorenog koda namijenjen skeniranju web poslužitelja koji je dizajniran za pronalazak nesigurnih datoteka, konfiguracija i programa bilo kojeg web poslužitelja. Provodi testiranje web poslužitelja na preko 6700 potencijalno opasnih programa ili datoteka, provjerava zastarjelost verzije više od 1250 poslužitelja i specifične probleme verzija više od 270 poslužitelja, provjerava konfiguracijske postavke poslužitelja te pokušava otkriti instalirane web poslužitelje i programe. Alat se nalazi pod GNU GPL licencom. Ažuriranja i dopune koda su dobrodošle [36].

3.5.1 Značajke alata Nikto

Od značajki se ističu [37]:

- Podržava HTTP Proxy
- Provjerava zastarjele komponente poslužitelja
- Podržava SSL transportni protokol
- Sprema izvješća u XML, CSV, HTML, tekstualnom i NBE formatu
- Jednostavno ažuriranje preko naredbene linije
- Podešavanje skeniranja da uključuje ili isključuje klase provjera ranjivosti
- Automatsko pauziranje u određenom vremenu

Nikto je alat izgrađena na LibWhisker2 Pearl biblioteci za kreiranje HTTP testnih skripti te se zbog toga može pokrenuti na svim platformama koje podržavaju Pearl okruženje. Može se jednostavno ažurirati preko naredbene linije [37].

3.5.2 Tumačenje statusnog koda različitih datoteka

Od Nikto verzije 2.0 alat više ne pretpostavlja da su stranice pogrešaka različitih tipova datoteka jednake, već se lista jedinstvenih datotečnih nastavaka generira iz testne baze podataka u vremenu izvođenja. Za različite datotečne nastavke mogu se razlikovati odgovori koji označavaju pogreške, na primjer odgovor „nije pronađen“ se razlikuje za .html i .cgi datoteku. Te prema tome za svaki tip datoteke postoji najbolja metoda određivanja pogrešaka. To može biti standardni RFC odgovor, poklapanje sadržaja ili MD4 hash. Zahvaljujući tome, Nikto može koristiti najbolju metodu za svaki pojedini tip datoteke i smanjiti broj negativno pozitivnih. U prošlim verzijama i kod nekih drugih sigurnosnih alata gdje se ne primjenjuje ova metoda, nego se uglavnom oslanjaju na HTTP odgovore za utvrđivanje postoji li neka stranica ili skripta, često

se pojavljuje slučaj da poslužitelji vraćaju odgovor 200 „OK“ za zahtjeve koji su zabranjeni ili zbog nekog razloga nisu pronađeni što na kraju vodi do krivog tumačenja i velikog broja lažno pozitivnih [37].

3.5.3 Preduvjet korištenja alata Nikto

Na svakom sustavu koji podržava instalaciju Pearl-a može se koristiti alat Nikto. Perl je skriptni jezik koji pohranjuje programe kao običan tekst koji se zatim pokreću u vremenu izvršavanja preko prevoditelja. Korištenje alata je testirano i moguće na Windows, Mac OSX i različitim Linux i Unix instalacijama [37]. Jedini Pearl modul koji nije standardno podržan je modul LibWhisker. Nikto je konfiguriran za korištenje lokalne LW.pm datoteke koja se nalazi u direktoriju dodataka. S Nikto 2.1.5 verzijom dolazi uključeni LibWhisker koji se malo razlikuje od standardnog [37]. Ukoliko SSL nije podržan potrebno je instalirati Net::SSLeay Perl modul. Za potrebe prijave na Metasploit moraju biti instalirani RPC::XML i RPC::XML::Client moduli. Iako Nikto može raditi bez tih modula, za više funkcionalnosti potrebna je navedena instalacija [37].

3.5.4 Sučelje alata Nikto

Po predefiniranim postavkama alat ne koristi grafičko sučelje, već se instalacija i rad odvijaju preko naredbene linije. Iako je to s jedne strane nedostatak, s druge strane korištenje CLI-ja (engl. Command Line Interface), je odlično za daljinsko pokretane alata preko SSH (engl. Secure Shell) veza.

Za korisnike kojima je važno grafičko sučelje, razvijen je NiktoQT. NiktQT je grafičko korisničko sučelje za Nikto skener. Korisniku nudi mogućnost unosa ciljane aplikacije, postavljanje dodatnih opcija, konfiguracijske datoteke, domaćina ili izlaza. Pritiskom na gumb pokreće se skeniranje, a rezultati skeniranja vidljivi su u radnom prozoru [38].

3.5.5 Rad alata Nikto

Nikto radi na način da šalje zahtjeve prema web poslužitelju i analizira odgovore. Koristi bazu podataka s URL adresama za provođenje zahtjeva za skeniranje. Poziva resurse koji upućuju na postojanje konfiguracije poslužitelja ili web aplikacije. Brzo otkriva greške kao što su izlistavanje direktorija i omogućene opcije uklanjanja pogrešaka, istražuje HTTPS verzije stranica te može biti podešen za skeniranje nestandardnih portova (npr. port 8080 na kojem slušaju mnogi Java web poslužitelji). Nikto može slati poslužitelju GET i POST podatke zajedno

sa zahtjevima te ispituje puni odgovor poslužitelja. Uz zadane datoteke, traži i nesigurne datoteke što može otkriti probleme kao što je problem zaboravljenih sigurnosnih kopija ili druge ranjivosti koje mogu ugroziti sigurnost poslužitelja [39].

3.5.6 Testiranje aplikacija pomoću alata Nikto

Za pokretanje skeniranja potrebno je upisati IP adresu ili naziv domaćina (engl. hostname). Prema predefiniranim postavkama radi na portu 80, a ukoliko se TCP port želi promijeniti potrebno je unijeti port nakon opcije „-p“. Osim IP adrese i naziva domaćina, željena aplikacija se može specificirati unosom URL adrese. Tim načinom, unosom URL adrese, se također mogu specificirati portovi, domaćini (engl. hosts) i protokoli. Za više kompleksna testiranja može se koristiti „-mutate“ parametar.

Nikto omogućuje testiranje na više portova i više domaćina. Za vrijeme istog skeniranja može se skenirati više portova na istom domaćinu na način da se nakon „-p“ opcije unese više portova odvojenih crticom (pr. 80-88) ili zarezom (pr. 80,88) čime se domaćin testira na svim navedenim portovima. Za vrijeme jedne sesije moguće je testirati više domaćina preko tekstne datoteke koja sadrži IP adrese ili imena domaćina (engl. host names). Nakon opcije „-h“ osim IP adrese ili naziva domaćina, može se postaviti naziv datoteke koja sadrži te podatke (IP adrese ili nazive domaćina) na način da se svaki domaćin ili IP adresa nalaze u svom retku, tj. po jedna adresa po retku datoteke, a na kraju svake se nalazi broj porta [37]. Proxy se može koristiti na dva načina: preko datoteke nikto.conf ili preko naredbene linije. Osim predefiniranih testova, korisnici mogu kreirati svoje vlastite testove koji će se učitati zajedno s ugrađenim provjerama.

Prema predefiniranim postavkama prikazuju se samo osnovne informacije o ranjivostima. Ako se želi dobit detaljniji pregled, potrebno je koristiti „-Display“ parametar. Izvještaj skeniranja moguće je pohraniti u raznim formatima: tekstualna datoteka, CSV, XML, HTML, NBE.

3.5.7 Otkrivanje ranjivosti korištenjem Nikto alata

Alat ispituje web poslužitelj kako bi pronašao pogrešne konfiguracije programskih datoteka i poslužitelja, zadane i nesigurne datoteke i programe te zastarjele programe i poslužitelje.

Prema [40], Nikto može otkriti sljedeće ranjivost

- Prijenos datoteka
- Zanimljive datoteke
- Pogrešna konfiguracija

- Nedozvoljeno objavljivanje informacija
- XSS ubrizgavanje
- HTML ubrizgavanje
- Ubrizgavanje skripte
- Dohvaćanje udaljenih datoteka
- Izvršavanje naredbi
- SQL ubrizgavanje
- Odbijanje izvršenja usluge
- Zaobilaženje prijave
- Uključivanje udaljenog izvora
- Identifikacija programske podrške

3.5.8 Kratki pregled i usporedba alata

U tablici 3.1. dan je kratak pregled svih korištenih alata. Tablična forma omogućuje jednostavnu usporedbu prethodno opisanih alata.

Tablica 3.1. prikazuje osnovne značajke opisanih alata za skeniranje. Osim navoda značajki, tablični prikaz pomaže u uočavanju razlika među alatima te usporedbi i odabiru odgovarajućeg alata s obzirom na korisničke potrebe.

Tab. 3.1. *Usporedba alata prema osnovnim značajkama*

	Arachni	**Vega**	**ZAP**	**W3af**	**Nikto**
Programski jezik	Ruby	Java	Java	Python	Perl
Besplatan za korištenje	Da (osim za slučaj komercijalizacije)	Da	Da	Da	Da
Izvorni kod	Dostupan	Dostupan	Dostupan	Dostupan	Dostupan
Licenca	Arachni Public Source License v1.0	Eclipse Public License v1.0	Apache 2.0	General Public License v2.0	General Public License
Autor	Tasos Laskos	Subgraph	OWASP.org	Andres Riancho	Chris Sullo i David Lodge
Način rada	Skener ili Proxy	Skener ili Proxy	Skener ili Proxy	Skener ili Proxy	Skener ili Proxy

Sučelje	Korisničko sučelje naredbene linije i web korisničko sučelje	Baziran na grafičkom korisničkom sučelju	Baziran na grafičkom korisničko m sučelju	Korisničko sučelje naredbene linije i grafičko korisničko sučelje	Baziran na sučelju naredbene linije
Proširivost	Da, preko dodataka	Da, preko JavaScript modula	Da, preko dodataka	Da, preko dodataka	Da, preko korisnički definiranih testova i dodataka
Operacijski sustavi	MS Windows, Mac OS X i Linux	Linux, Mac OS i MS Windows	Linux, MS Windows i Mac OS X	Linux, Mac OSX, kompatibilan s MS Window-sima	MS Windows, Mac OSX i Linux

Alati koji se mogu koristiti u prevenciji pojedinih ranjivosti koje se pojavljuju u „OWASP Top 10“ listi za 2017. godinu:

1. Ubrizgavanje (engl. injection) – Arachni, Vega, ZAP, w3af, Nikto
2. Neispravna autentifikacija i upravljanje sjednicama (engl. broken authentication) – Arachni, ZAP, w3af
3. Izlaganje osjetljivih podataka (engl. sensitive data exposure) – Arachni, Vega, ZAP
4. XXE (engl. XML eXternal Entity) – w3af
5. Loša kontrola pristupa (engl. broken access control) – ZAP (Jednostavno otkrivanje s ručnim testiranjem, teško se otkriva sa statičkim ili dinamičkim skeniranjem.)
6. Pogrešna konfiguracija sigurnosti (engl. security misconfiguration) – Arachni, w3af
7. XSS (engl. Cross-Site Scripting) – Arachni, Vega, ZAP, w3af, Nikto
8. Nesigurna deserijalizacija (engl. insecure deserialization) - ZAP
9. Korištenje komponenti s poznatim ranjivostima (engl. vulnerable components) – Nikto (provjerava zastarjele komponente)
10. Nedovoljno praćenje i bilježenje (engl. insufficient logging and monitoring) - ZAP

S obzirom na „OWASP Top 10“ listu objavljenu za 2017. godinu, u ovom poglavlju je naveden primjer korištenja pojedinih alata po pojedinim ranjivostima objavljenima za 2017. godinu kako bi se otkrile i bile obuhvaćene sve najčešće ranjivosti koje se spominju u listi. Osim navedenih ranjivosti analizirani alati mogu otkriti veliki broj drugih ranjivosti koje se ne spominju u navedenoj OWASP Top 10 listi.

4 IMPLEMENTACIJA I ANALIZA SIGURNOSNIH ALATA

Alati za sigurnosno skeniranje web aplikacija su programi koji komuniciraju s web aplikacijom preko pristupnog dijela (engl. front-end) kako bi identificirali potencijalne ranjivosti i slabosti arhitekture koje se mogu pojaviti. Glavna zadaća sigurnosnih skenera je otkrivanje sigurnosnih propusta i na taj način olakšavanje automatiziranih preglede web aplikacija. Napravljeni su tako da traže različite prijetnje, tj. ranjivosti kao što su: validacija ulazno/izlaznih podataka, specifične probleme aplikacije ili pogreške u konfiguraciji servera. Za razliku od testiranja, tj. skeniranja izvornog koda, skeneri web aplikacija nemaju pristup izvornom kodu već se ranjivosti i prijetnje detektiraju izvršavanjem stvarnih napada. Budući da nemaju pristup izvornom kodu radi se o testiranju po principu crne kutije (engl. black box testing).

Alati za skeniranje web aplikacija i pronalazak ranjivosti se mogu testirati i koristiti na raznim aplikacijama. Za potrebe testiranja, u sljedećim potpoglavljima analizirani alati se testiraju na web aplikaciji „Kuharica“. Aplikacija je napravljena korištenjem PHP i JavaScript programskog jezika, HTML-a, CSS-a i phpMyAdmin-a za MySQL bazu podataka. Aplikacija omogućuje korisniku, nakon prijave u sustav ili registracije, postavljanje vlastitog recepta koji može biti dostupan svima ili privatan. Recepte koji su javni mogu svi vidjeti. Korisnici prijavljeni u sustav mogu vidjeti cijeli recept zajedno s postupkom pripreme, a korisnici koji nisu prijavljeni u sustav samo naziv i sliku.

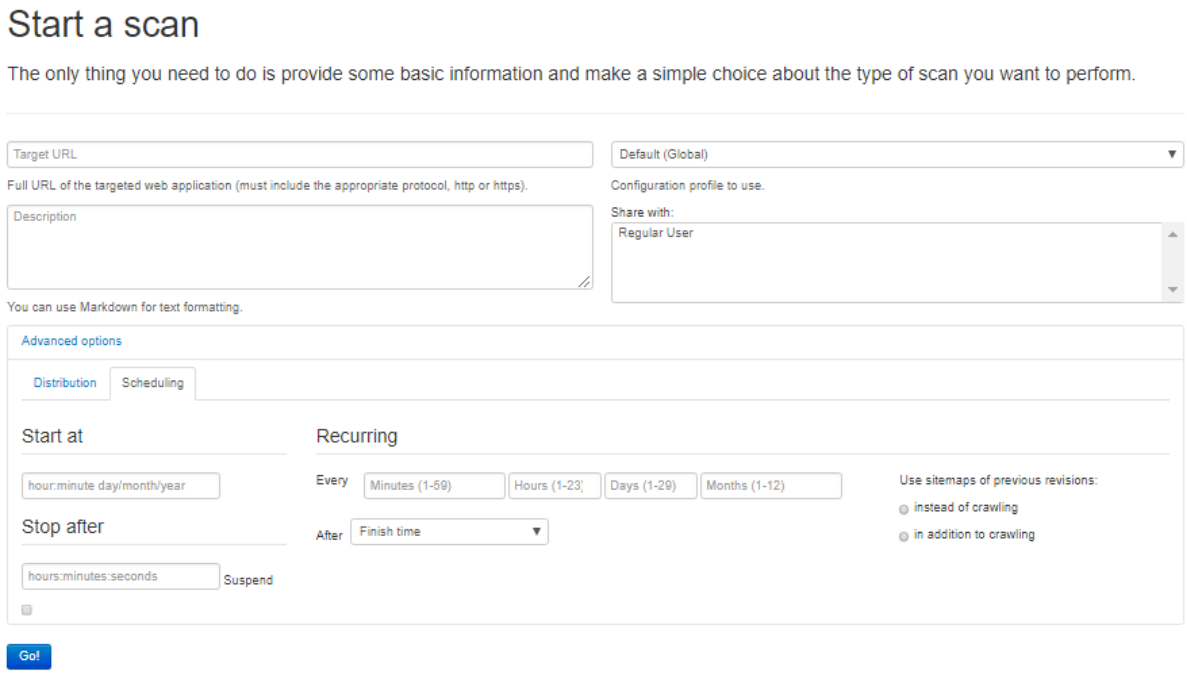
4.1 Implementacija i analiza alata Arachni

Arachni je Ruby okvir koji omogućuje jednostavno ispitivanje sigurnosti web aplikacija. Podržava rad na MS Windows, Mac OS X i Linux-u. Pruža podršku za JavaScript/DOM/HTML5/AJAX, kao i dodatnu optimizacija za česte JavaScript okvire kao što je AngularJS, JQuery. Budući da se svaki resurs aplikacije analizira pojedinačno, omogućena je prilagođavanja zahtjeva tehnologijama koje se koriste [18].

Zahvaljujući sposobnosti treniranja Arachni uči tijekom cijelog postupka skeniranja koje izvodi. Uči na HTTP odgovorima tijekom procesa praćenja i provjeravanja izvješća kako bi mogao identificirati nove ulazne vektore i lažne rezultate te upravljati složenim tijekom rada.

4.1.1 Preuzimanje i pokretanje Arachni skeniranja

Arachni alat za skeniranje web aplikacija je dostupan za preuzimanje na sljedećoj adresi: [41], a izvorni kod skenera se nalazi na sljedećoj adresi: [42]. Nakon preuzimanja i pokretanja alata prvo se potrebno prijaviti u sustav koristeći neki od predefiniranih korisničkih podataka koji se nalaze u README datoteci. Kada je korisnik prijavljen u sustav može pokrenuti skeniranje. Za početak skeniranja potrebno je unijeti par osnovnih informacija i kliknuti na gumb „Go!“ kako bi skeniranje započelo. Izgled prozora za postavljanje parametara skeniranja prikazan je na slici 4.1.



Sl. 4.1. *Postavljanje početnih parametara*

Osim URL adrese stranice koju se želi skenirati, korisnik može unijeti vrijeme skeniranja aplikacije. Ukupno vrijeme skeniranja aplikacije Kuharica postavljeno je na 2 sata, nakon čega aplikacija prestaje sa skeniranjem te se korisniku nude rezultati skeniranja kao i dogovarajući izvještaji. Korisnik proizvoljno može postaviti neko drugo vrijeme trajanja testiranja kao i vrijeme kada želi da skeniranje započne.

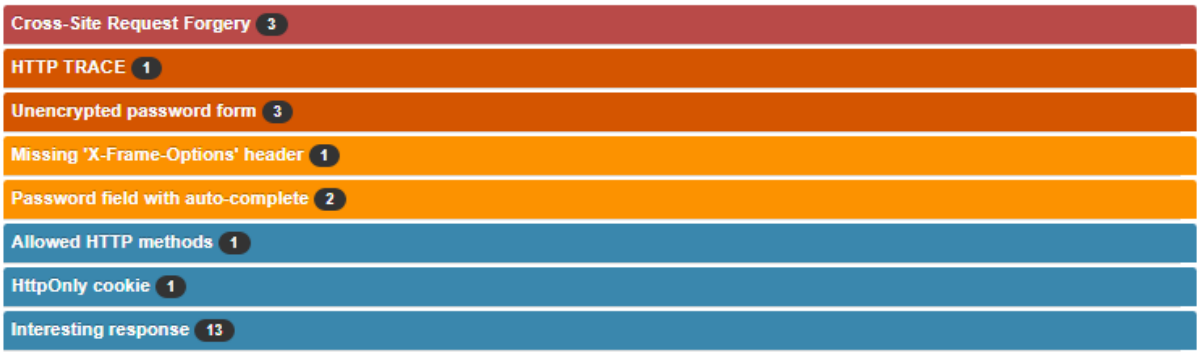
4.1.2 Prijavljene greške po jakosti i tipu nakon Arachni skeniranja

Prikaz prijavljenih grešaka po jakosti dan je na slici 4.2.

Sl. 4.2. *Prijavljene greške po jakosti*

Jakost odnosno, rizik pojedine ranjivosti predstavlja visinu štete koju ranjivost može uzrokovati. S obzirom na dobivene rezultate imaju tri visoka rizika, četiri srednja i tri niska rizika te petnaest prijava informativnog sadržaja. Rizici su vizualno prikazani različitim bojama: tamno crvena za viske rizike, crvena za srednje, narančasta za niske rizike i plava za informacije o greškama koje nemaju veliki utjecaj na sigurnost aplikacije.

Prikaz prijavljenih grešaka po tipu dan je na slici 4.3.

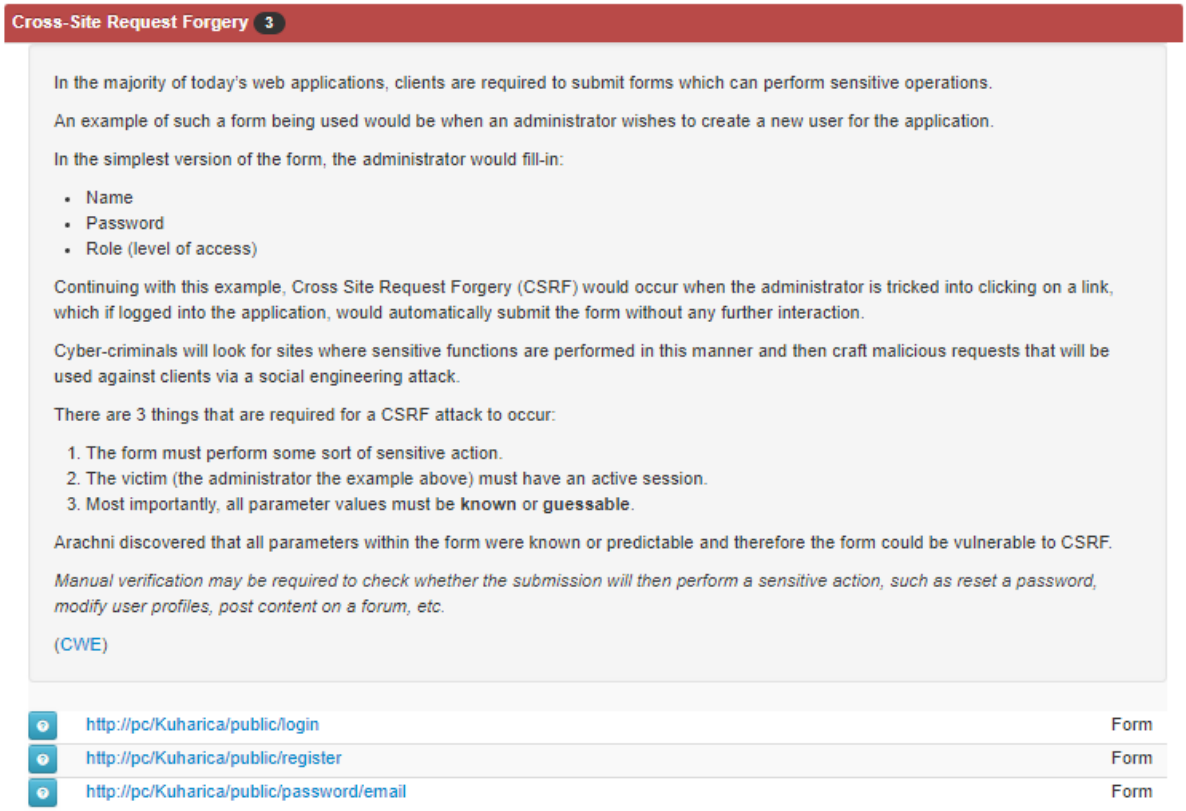


Sl. 4.3. *Prijavljene greške po tipu*

Prema tipu, alat je pronašao tri CSRF ranjivosti koje spadaju u visoko rizične ranjivosti, zatim jednu HTTP TRACE (prikazuje vraćeni unos natrag korisniku) i tri ranjivosti vezane uz nešifrirane lozinke koje pripadaju skupini srednjih rizika, nadalje jednu ranjivost o nedostatku zaglavlja „X-Frame-Options“ i dvije ranjivosti automatskog nadopunjavanja lozinki koje pripadaju skupini nisko rizični ranjivosti i na posljetku prikazuje informacije o greškama koje nisu rizične (u ovom primjeru: dozvoljene HTTP metode, HttpOnly kolačić i zanimljivi odgovori). Kao i kod prikaza rizika ranjivosti (visok, srednji, nizak) i u ovom prikazu različiti rizici prikazani su različitim bojama.

4.1.3 Pojašnjenje prijavljenih ranjivosti nakon Arachni skeniranja

Primjer objašnjenja prikazan je slikom 4.4.

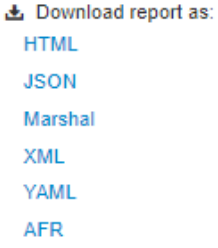


Sl. 4.4. *Prikaz pojašnjenja ranjivosti*

Na slici 4.4 prikazano je pojašnjenje CSRF ranjivosti. Dan je slučaj izvođenja CSRF napada u kojem administrator koji ispunjava obrazac s osjetljivim podacima klika na poveznicu i na taj način omogućuje napadačima djelovanje. Prema dobivenom objašnjenju kako bi napad bio uspješan moraju biti zadovoljena tri uvjeta: obrazac treba sadržavati osjetljive podatke (izvođenje osjetljivih akcija), treba biti aktivna sjednica i sve vrijednosti parametara trebaju biti poznate (ili ih je moguće pogoditi). Alat Arachni je utvrdio da su tri obrasca aplikacije osjetljiva na ovu vrstu napada, tj. da su poznati ili predvidljivi parametri unutar forme.

4.1.4 Mogućnosti preuzimanja Arachni izvještaja

Arachni također omogućuje preuzimanje izvještaja u raznim oblicima:



Sl. 4.5. *Mogućnosti preuzimanja izvještaja*

Arachni skener dozvoljava preuzimanje i pregled izvješća skeniranja u različitim formatima. Format izvješća može biti: html, xml, json, text, marshal, yaml, afr (engl. Arachni Framework Report). Afr je Arachni izvješće koje se može pretvoriti u bilo koji od navedenih formata.

4.1.5 Prijava ranjivosti alata Arachni s obzirom na OWASP Top 10

Prijavljene greške s obzirom na OWASP Top 10:

- A5-Pogrešna konfiguracija sigurnosti (engl. security misconfiguration) – pronađena je 1 ranjivost, uzrok: HTTP TRACE

Vector type	HTTP method	Action
server	TRACE	http://pc/Kuharica/public/

Sl. 4.6. *Ranjivost - Pogrešna konfiguracija sigurnosti*

Tijekom skeniranja pronađena je jedna ranjivost koja odgovara jednoj točki s OWASP liste ranjivosti. Radi se o pogrešnoj konfiguraciji sigurnosti koju, u ovom slučaju, uzrokuje HTTP TRACE metoda.

- A8-Izloženost osjetljivih podataka (engl. sensitive data exposure) – pronađene su tri ranjivosti, uzrok: CSRF (engl. Cross-Site Request Forgery):

Vector type	HTTP method	Action
form	GET	http://pc/Kuharica/public/login
form	GET	http://pc/Kuharica/public/register
form	GET	http://pc/Kuharica/public/password/email

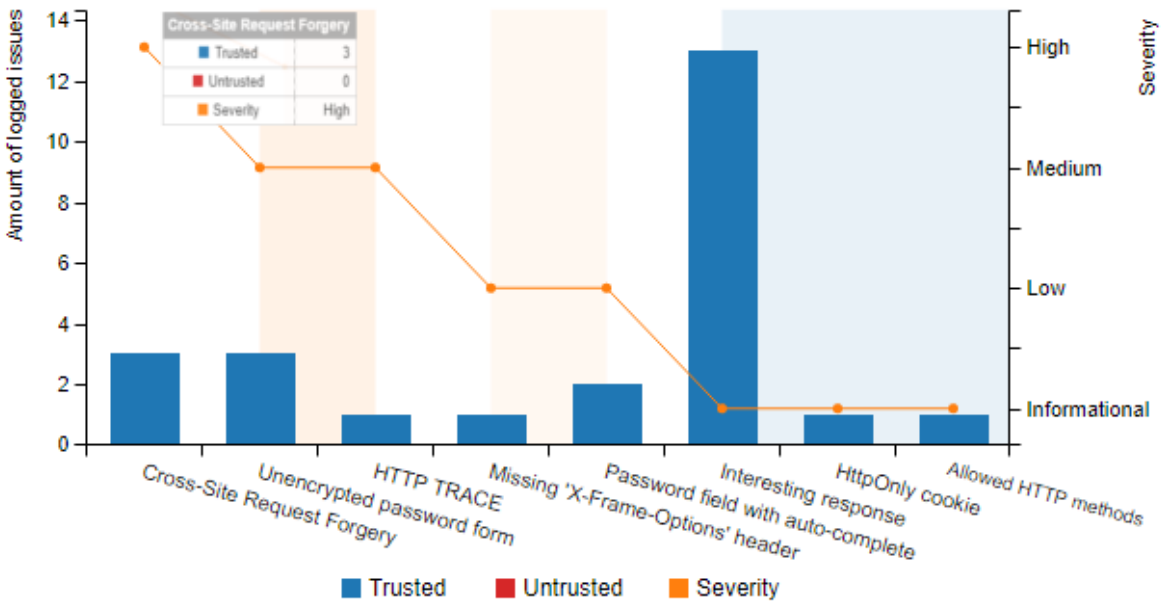
Sl. 4.7. *Ranjivost - Izlaganje osjetljivih podataka*

Na OWASP listi ranjivosti nalazi se „Izloženost osjetljivih podataka“. U skeniranoj aplikaciji alat je pronašao tri ranjivosti koje odgovaraju spomenutoj ranjivosti s OWASP liste. Radi se o CSRF ranjivostima preko kojih može doći do izlaganja korisničkih podataka preko forme.

4.1.6 HTML izvještaj nakon Arachni skeniranja

Alat nudi grafički prikaz pronađenih ranjivosti u HTML obliku.

- Prikaz potencijalnih opasnosti prema tipu, povjerljivosti i opasnosti prikazan je na slici 4.8.



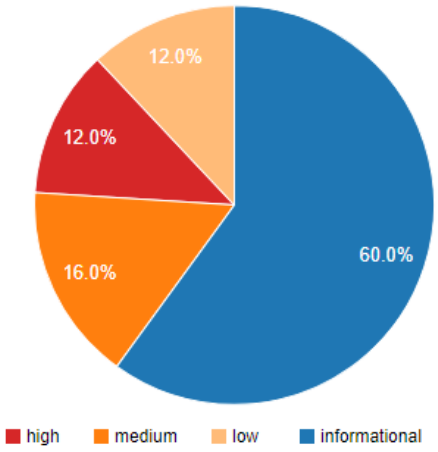
Sl. 4.8. *Grafički prikaz potencijalnih opasnosti*

Grafikonom su prikazani pronađeni potencijalni rizici prema:

- Tipu - CSRF, HTTP TRACE, HttpOnly kolačić i dr.,
- Povjerljivosti - povjerljivo (engl. trusted), nepovjerljivo (engl. untrusted),
- Opasnosti – visoka, srednja, niska, informativna.

Pozicioniranjem miša na narančastu točku na liniji prikazuju se iznad navedene informacije za svaki tip ranjivosti. Otvoreni „Cross-Site Request Forgery“ prozorčić osim naziva prikazuje da se radi o povjerljivim i visoko ozbiljnim ranjivostima.

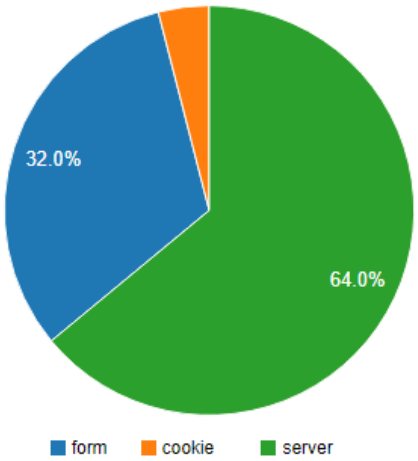
- Opasnosti s obzirom na potencijalne napade, tj. rizike koje napadi mogu prouzrokovati prikazane su na slici 4.9.



Sl. 4.9. *Rezultati skeniranja prikazani prema opasnostima napada*

Prema dobivenom izvještaju skeniranja najviše je prijavljeno informativnih rizika (60%) koji nemaju veliki utjecaj na sigurnost aplikacije. Nakon njih po broju prijavljenih rizika slijede rizici srednje jakosti (16%) dok visokorizičnih i niskorizičnih grešaka ima jednako (12%).

- Elementi preko kojih se mogu izvršiti napadi:



Sl. 4.10. *Rezultati testiranja s obzirom na elemente preko kojih se mogu izvršiti napadi*

Većina napada na aplikaciju koja sadrži pronađene rizike se može očekivati sa strane server, čak 64% dok se duplo manje napada može očekivati preko forme (32%) i najmanji broj napada preko kolačića.

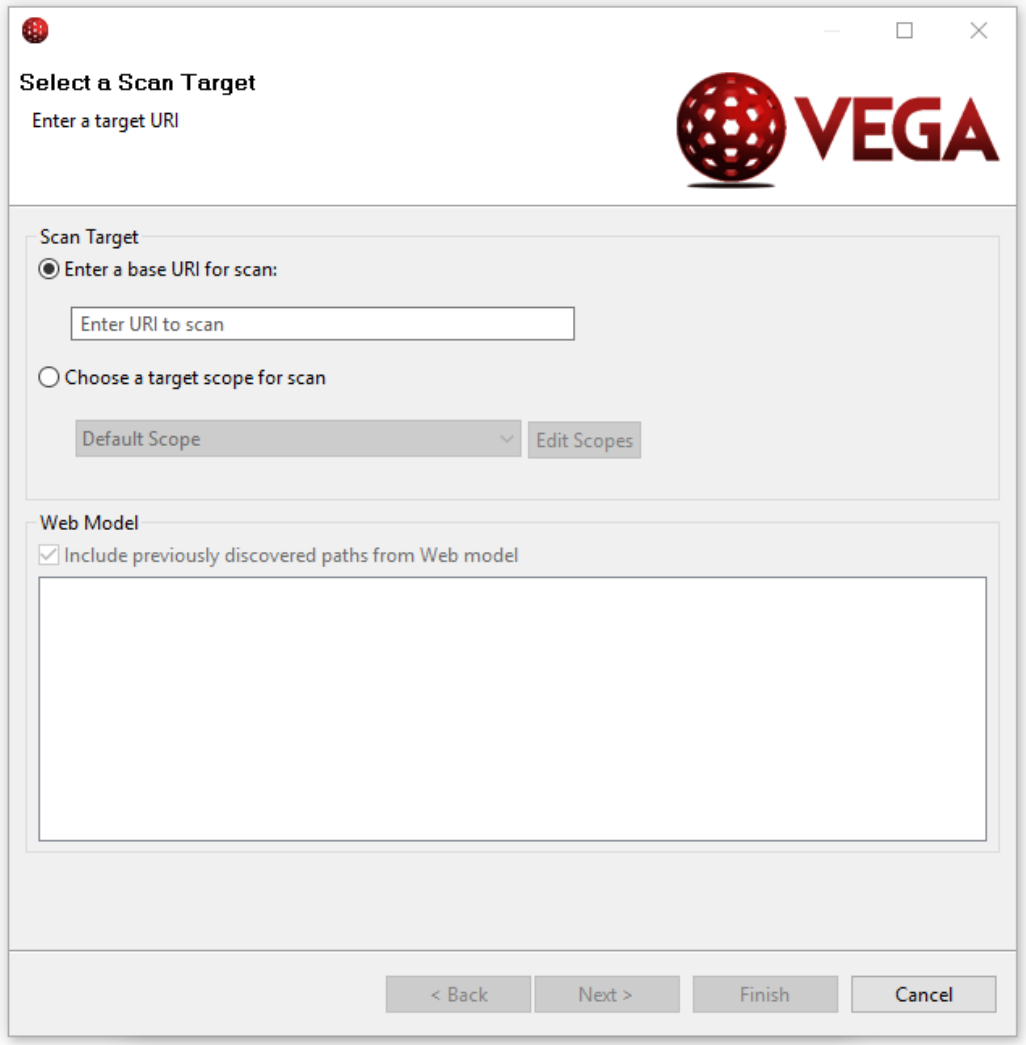
4.2 Implementacija i analiza alata Vega

Vega je besplatan sigurnosni skener web aplikacija. Napisan je u Javi i može se pokrenuti na Windows, Linux i OS X [22].

4.2.1 Preuzimanje i pokretanje alata Vega

Alat Vega za skeniranje web aplikacija je dostupan za preuzimanje na sljedećoj adresi: [43], a izvorni kod skenera se nalazi na sljedećoj adresi: [44]. Nakon preuzimanja i pokretanja alata može se započeti sa postavljanjem parametara skeniranja, a potom i sa samim skeniranjem.

Prozor za postavljanje parametara skeniranja:



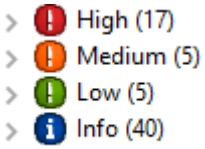
Sl. 4.11. *Početak skeniranja, postavljanje parametara*

Za skeniranje je potrebno upisati URI aplikacije koju je potrebno skenirati i kliknuti na „Next“ gumb. Nakon postavljanja URI-ja aplikacije skeniranje može započeti.

4.2.2 Rezultati Vega skeniranja s obzirom na jakost ranjivosti

Testiranje je ukupno trajalo 5 minuta.

Prikaz rezultata skeniranja s obzirom na jakost prijetnje, tj. opasnost ranjivosti:



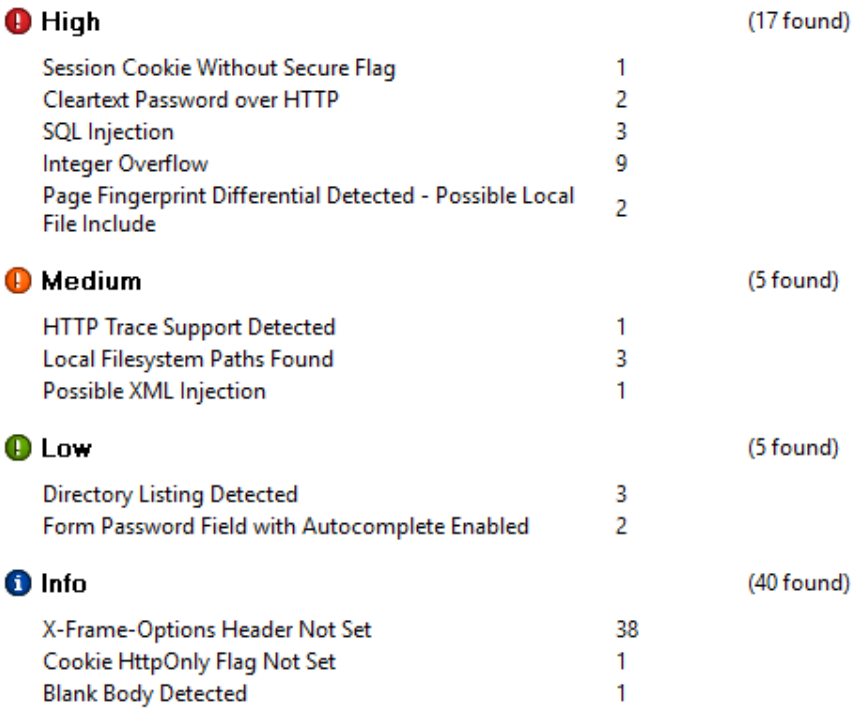
Sl. 4.12. *Rezultati skeniranja s obzirom na opasnost ranjivosti*

Prema dobivenim rezultatima skeniranja najviše ima informativnih prijava (40). Nakon informativnih slijede visoko rizične ranjivosti (17) te srednjerizične i niskorizične ranjivosti (5). Opasnosti pojedinih ranjivosti su prikazane različitim bojama: crvena za visokorizične, narančasta za srednjerizične, zelena za niskorizične ranjivosti i plava za informativne prijave koje nemaju veliki utjecaj na sigurnost aplikacije.

4.2.3 Prikaz tipova ranjivosti nakon Vega skeniranja

Prikaz jakosti ranjivosti može se razgranati tako da se za svaku jakost, odnosno rizik ranjivosti prikažu pronađeni tipovi ranjivosti.

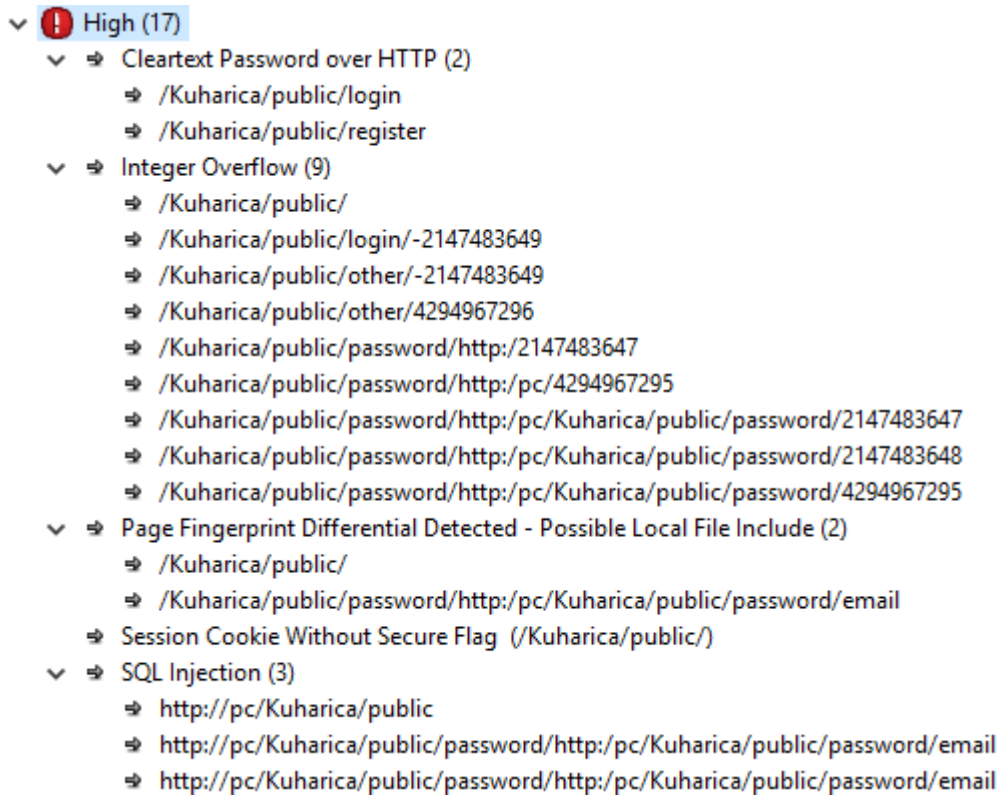
Prikaz prijavljenih ranjivosti po tipu prikazan je na slici 4.13.



Sl. 4.13. *Rezultati skeniranja s obzirom na opasnost ranjivosti*

Osim težine rizika nudi se prikaz razgranatog stabla, odnosno tipova ranjivosti prema rizicima. Kod visokog rizika pojavljuje se šest tipova ranjivosti od kojih je najzastupljenija „Integer Overflow“ sa devet pojavljivanja, a najmanje zastupljena „Session Cookie Without Secure Flag“ koja se pojavljuje samo jednom. Za srednje rizične ranjivosti otkriveno je tri tipa: „HTTP Trace Support Detect“ (1) i „Possible XML Injection“ (1) te „Local Filesystem Paths Found“ (3). Pronađena su samo dva tipa koja pripadaju skupini srednjerizičnih ranjivosti, a to su izlistavanje direktorija (3) i automatsko nadopunjavanje unosa za lozinku u formi (2). Od informativnih prijava najzastupljenije je ne postavljeno X-Frame-Options zaglavlje (38), a uz to se još po jednom pojavljuju prijave detekcije praznog tijela (engl. blank body detected) i nepostavljen HttpOnly kolačić.

Razgranato stablo ranjivosti visokog stupnja rizika prikazano je na slici 4.14.



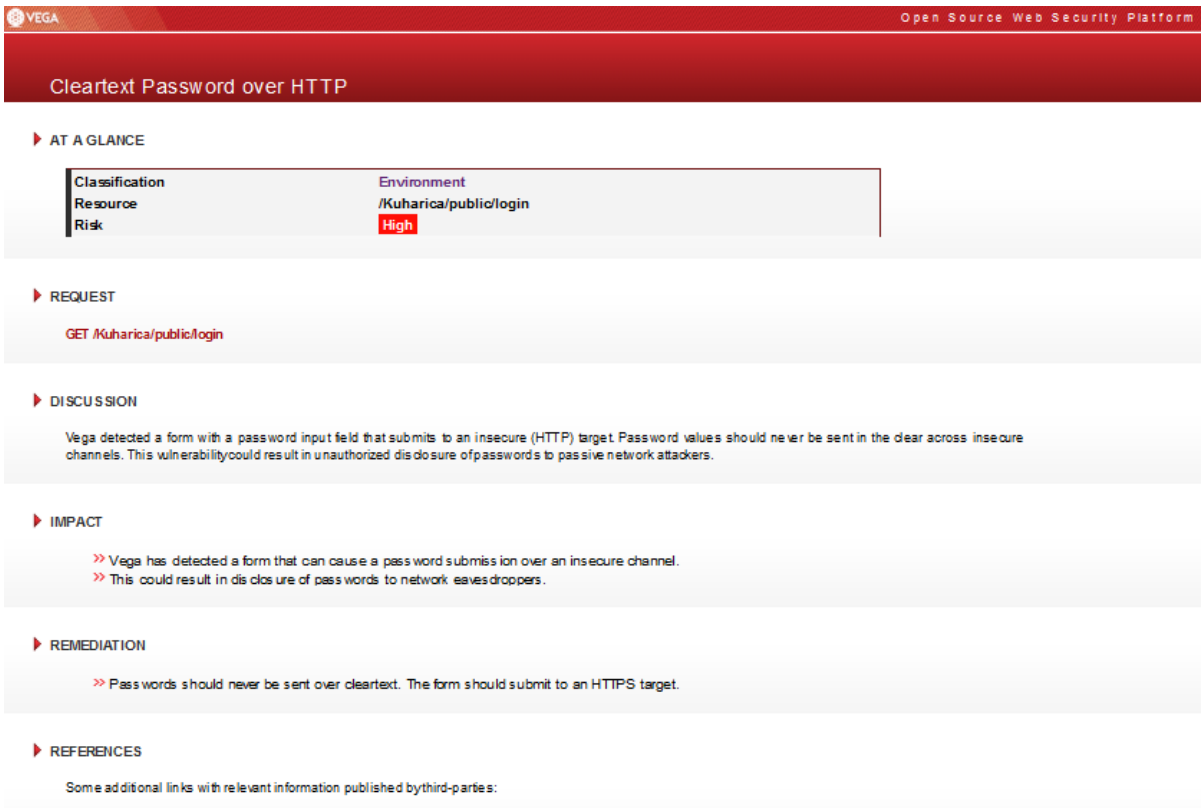
Sl. 4.14. *Razgranato stablo ranjivosti*

Stablo s prikazom stupnja rizika i odgovarajućim tipovima može se još razgranati tako da prikazuje gdje se sve pojavljuju pronađene ranjivosti. Na slici 4.14. prikazano je razgranato stablo visokorizičnih ranjivosti prema tipu i mjestu detekcije, odnosno pojavljivanja ranjivosti.

4.2.4 Pojašnjenje ranjivosti nakon Vega skeniranja

Na slici 4.15. prikazan je primjer pojašnjenja ranjivosti.

Osim prikaza ranjivosti moguće je pogledati pojašnjenje pojedine ranjivosti. U pojašnjenju se prikazuju zahtjev koji je poslan aplikaciji, rizik pronađene ranjivosti, diskusija u kojoj se navodi na temelju čega je pronađena i prijavljena ranjivost (u ovom slučaju radi se o obrascu koji sadrži unos lozinke koja se nakon potvrde unosa nesigurno šalje), utjecaju pronađene ranjivosti i savjetom za poboljšanje te dodatne poveznice s objašnjenjem ranjivosti (ukoliko postoje).



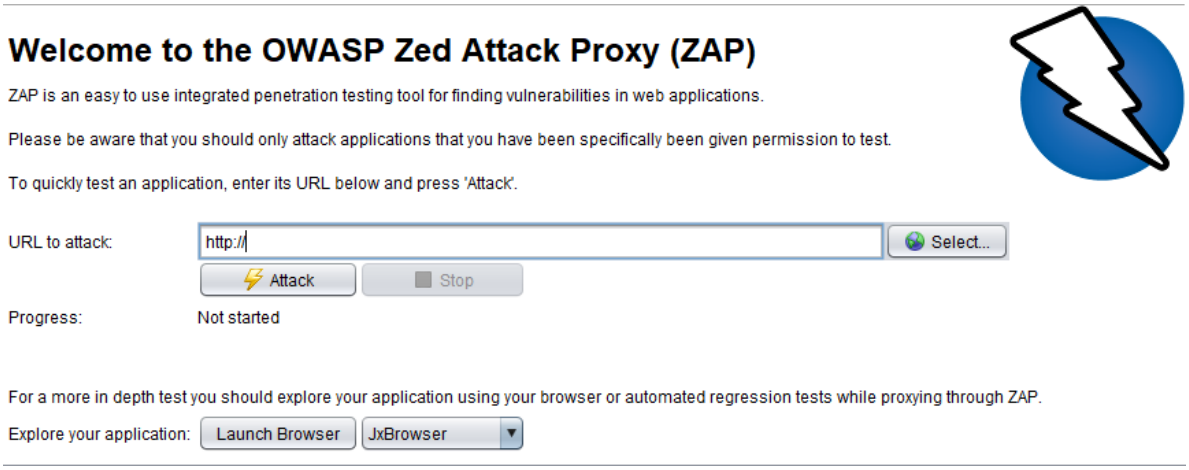
Sl. 4.15. *Pojašnjenje ranjivosti*

4.3 Implementacija i analiza alata OWASP ZAP

ZAP (engl. Zed Attack Proxy) je alat otvorenog koda koji služi za skeniranje web aplikacija. Napisan je u Javi i dostupan je za MS Windows, Linux i Mac OS X operacijske sustave [26].

4.3.1 Preuzimanje i pokretanje alata ZAP

ZAP alat za skeniranje web aplikacija je dostupan za preuzimanje na sljedećoj adresi: [45], a izvorni kod skenera se nalazi na sljedećoj adresi: [46]. Nakon preuzimanja ZAP alata i pokretanja korisniku se otvara prozor za postavljanje parametara skeniranja od kojih je najvažniji unos URL-a koji se želi napasti. Kada je URL unesen može se započeti s automatskim skeniranjem klikom na gumb „Attack“.



Sl. 4.16. *Postavljanje parametara skeniranja*

ZAP nudi više mogućnosti testiranja. Osim pasivnog skeniranja postoji i aktivno skeniranje. Quick Start, tj. Spider testiranje je trajalo oko 8 minuta, a Active Scan testiranje oko 1 minutu.

4.3.2 Prijavljene razine rizika nakon završetka ZAP skeniranja

Nakon testiranja dobiveni su sljedeći rezultati s obzirom na razinu rizika:

Sl. 4.17. *Rezultati skeniranja s obzirom na rizik pronađenih tipova ranjivosti*

Ukupno je pronađeno deset tipova ranjivosti s dva stupnja upozorenja. Zastavice koje se razlikuju po bojama označavaju različitu rizičnost ranjivosti. Rizičnost ranjivosti po bojama prikazana je na slici 4.18.

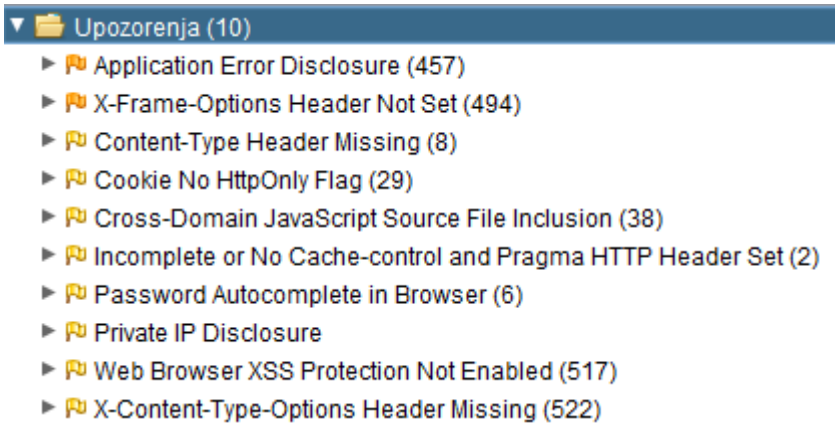
	Visoka (engl. High)
	Srednja (engl. Medium)
	Niska (engl. Low)
	Informacijska(engl. Informational)

Sl. 4.18. *Pojašnjenje stupnjeva rizika*

Postoje četiri stupnja rizičnosti: visoki stupanj označava da se radi o ranjivosti s velikim sigurnosnim rizikom, srednji stupanj da se radi o srednje rizičnoj ranjivosti, niski stupanj o ranjivosti koja nema veliki utjecaj na sigurnost aplikacije te na kraju informacijske prijave rizika koji prema procjeni skenera ne utječu na sigurnost.

4.3.3 Razgranato stablo ranjivosti nakon ZAP skeniranja

Alat ZAP nudi korisniku mogućnost prikaza stabla ranjivosti, odnosno upozorenja koja je alat pronašao. Stablo ranjivosti prikazano je na slici 4.19.

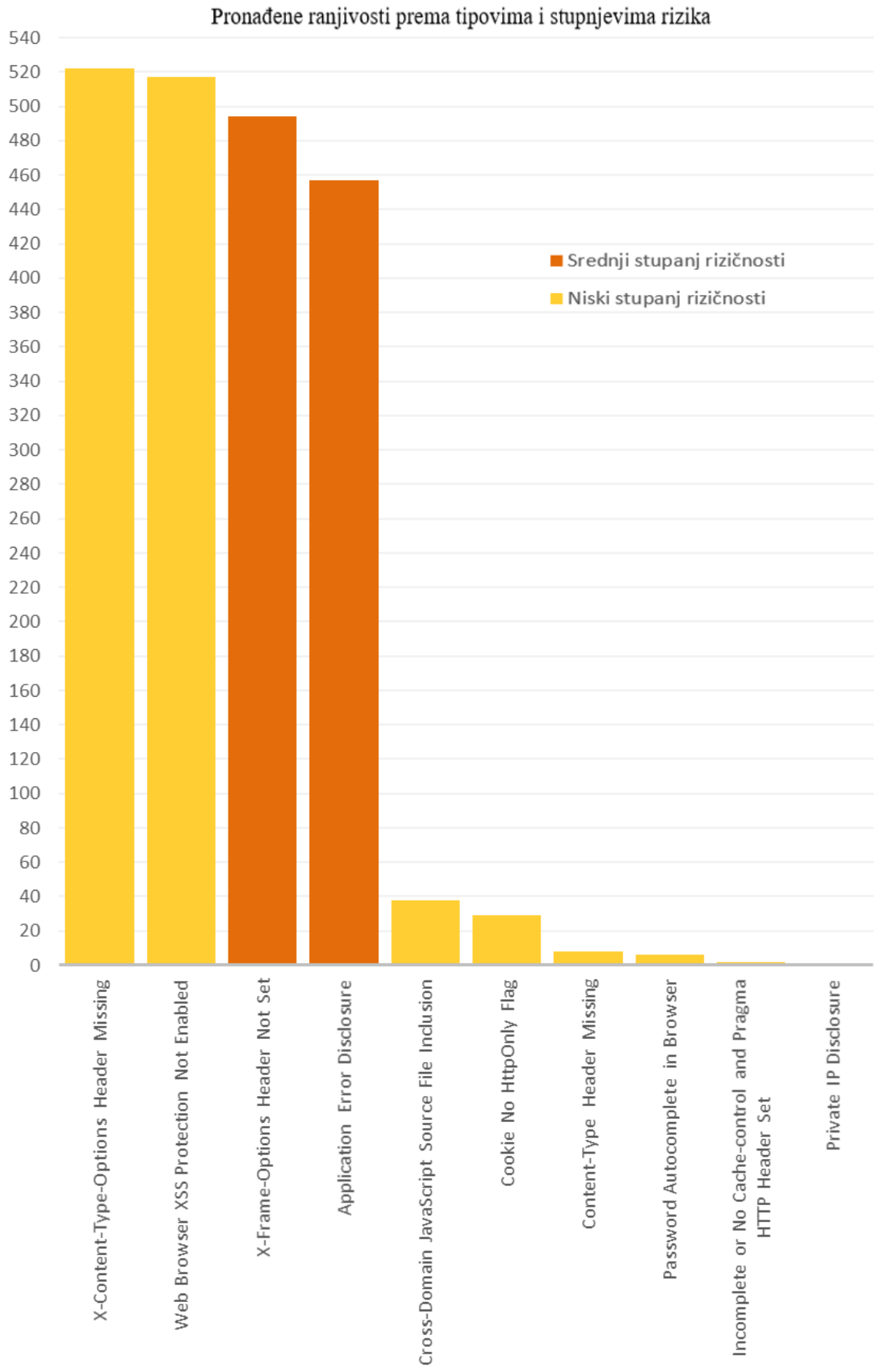


Sl. 4.19. *Prikaz stabla ranjivosti*

Na slici 4.19 prikazano je razgranato stablo svih prijavljenih upozorenja. Dvije narančaste zastavice označavaju upozorenja srednjeg stupnja (dva tipa ranjivosti), a preostale žute zastavice upozorenja niskog stupnja rizičnosti (osam tipova ranjivosti).

4.3.4 Izvještaj alata ZAP o pronađenim ranjivostima

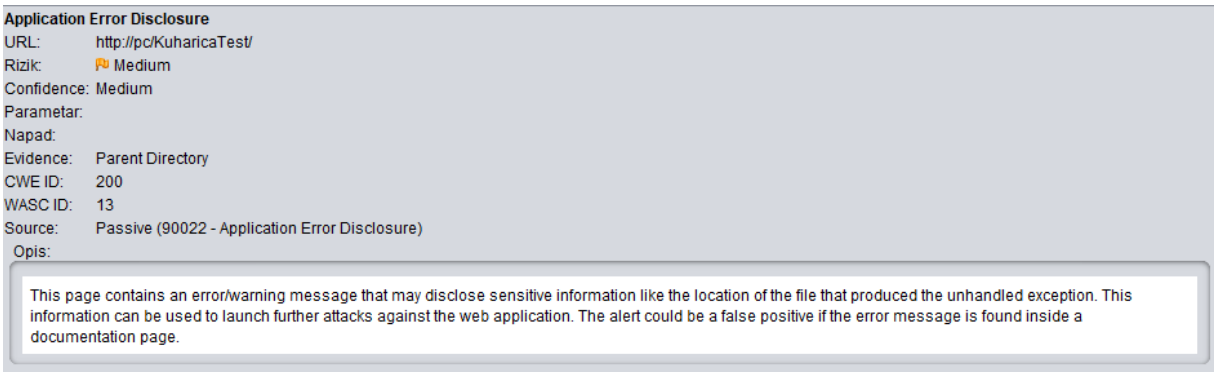
S obzirom na generirano izvješće pronađeno je dva tipa ranjivosti srednjeg stupnja rizičnost i osam tipova ranjivosti nižeg stupnja rizičnosti. Osim prema broju pronađenih tipova ranjivosti, ukupan broj ranjivosti koje odgovaraju pojedinim tipovima je također veći kod nižeg stupnja ranjivosti kao što se može vidjeti na slici 4.20. Narančastom bojom prikazani su rizici srednjeg stupnja rizičnosti, a žutom niskog stupnja rizičnosti.



Sl. 4.20. *Pronađene ranjivosti prema tipu i riziku*

4.3.5 Pojašnjenje ranjivosti nakon ZAP skeniranja

Primjer pojašnjenja ranjivosti:



Sl. 4.20. *Pojašnjenje ranjivosti*

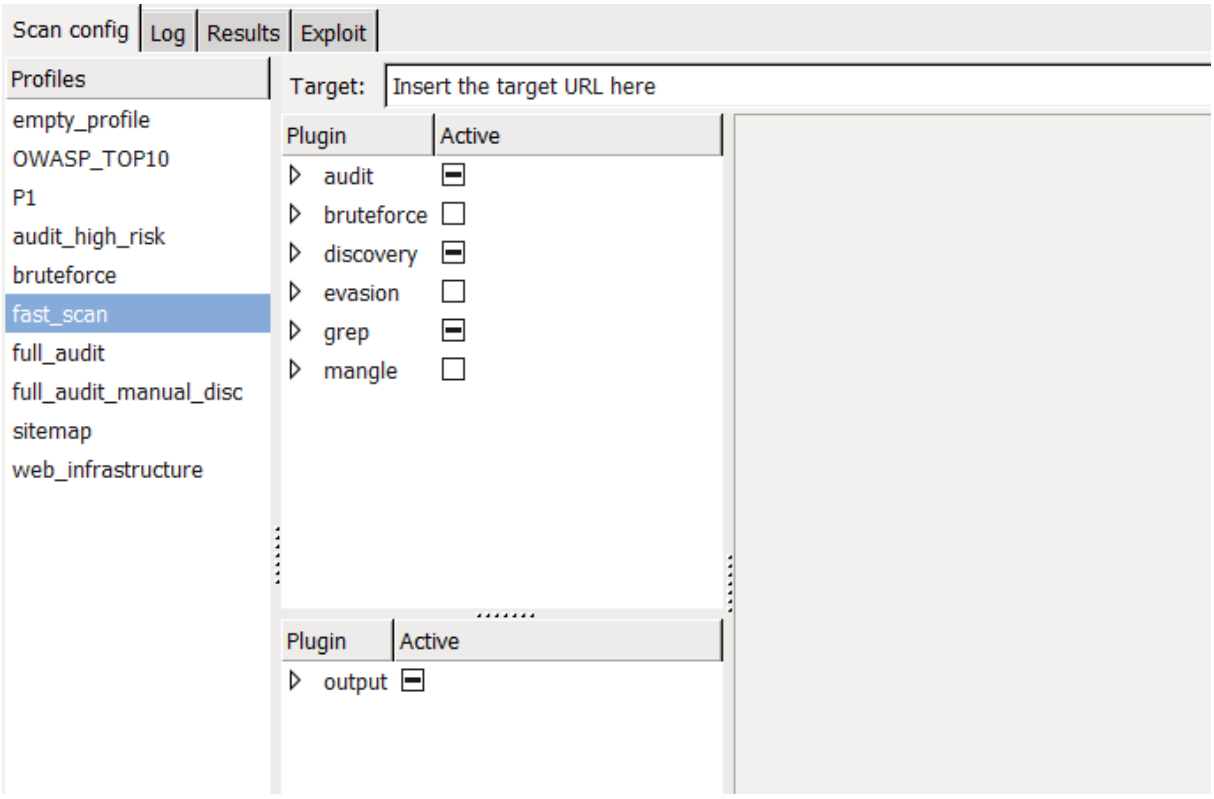
Pojašnjenje ranjivosti se sastoji od navoda činjenica kao što su URL i rizik te opisa ranjivosti. Opis obuhvaća pojašnjenje uzroka ranjivosti kao i posljedice koja može nastati. Za konkretan primjer opisa, na slici 4.21, navodi se da može doći do izlaganja osjetljivih podataka kao što je lokacija datoteke preko poruke o pogreški koja može uzrokovati daljnje napade.

4.4 Implementacija i analiza alata W3af

W3af (kratica od engl. Web Application Attack and Audit Framework) je alat otvorenog koda namijenjen za sigurnosno skeniranje web aplikacija. Razvijen je koristeći Python te podržan na sljedećim operacijskim sustavima: Windows, OS X, Linux, FreeBSD, OpenBSD [34].

4.4.1 Preuzimanje i pokretanje w3af skeniranja

W3af alat za skeniranje web aplikacija je dostupan za preuzimanje na sljedećoj adresi: [47], a izvorni kod skenera se nalazi na sljedećoj adresi: [48]. Nakon preuzimanja i pokretanja, za skeniranje je potrebno unijeti odgovarajuću URL adresu koja se želi skenirati, odabrati vrstu skeniranja i potom kliknuti na start kako bi započelo skeniranje. Prozor alata w3af s opcijama testiranja prikazan je na slici 4.22.



Sl. 4.212. *Početna stranica testiranja*

Od opcija skeniranja ističu se:

- fast_scan

  Opis skeniranja:

  Perform a fast scan of the target site, using only a few discovery plugins and the fastest audit plugins.

  Sl. 4.223. *Opis vrste skeniranja*

  Koristi se samo nekoliko dodataka za pretraživanje i najbrži dodaci za provjeru.

- full_audit scan

  Opis skeniranja:

  This profile performs a full audit of the target website, using only the webSpider plugin for discovery.

  Sl. 4.234. *Opis vrste skeniranja*

Koristi se samo webSpider dodatak za pretraživanje i potpuna provjera ciljane web stranice.

4.4.2 Trajanje skeniranja i broj otkrivenih ranjivosti nakon w3af skeniranja

Testiranje je provedeno korištenjem full_audit opcije skeniranja.

- Trajanje skeniranja:

Sl. 4.245. *Vrijeme trajanja skeniranja*

Skeniranje je ukupno trajalo 5 sati, 49 minuta i 55 sekundi. Nakon završetka skeniranja dostupan je potpuni izvještaj sa svim pronađenim ranjivostima.

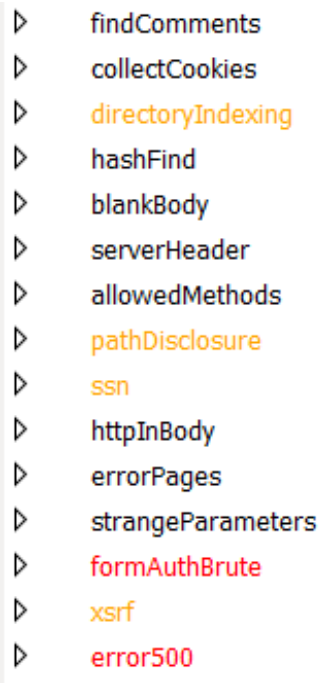
- Broj otkrivenih potencijalnih opasnosti:

Sl. 4.256. *Rezultati skeniranja s obzirom na rizik*

Otkriveno je 4356 informativnih ranjivosti i 1793 zabrinjavajućih ranjivosti.

4.4.3 Prikaz potencijalnih opasnosti u w3af izvješću

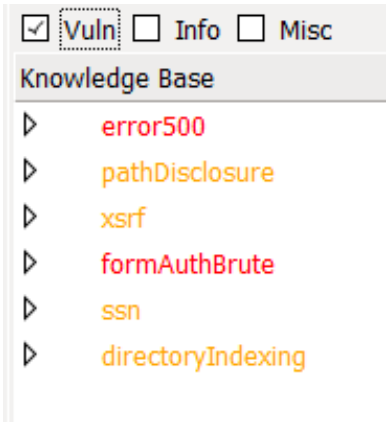
Potencijalne opasnosti su prikazane na slici 4.27.



Sl. 4.267. *Rezultati testiranja s obzirom na potencijalne opasnosti*

Na slici 4.27. prikazane su sve prijave: potencijalne ranjivosti (6) – prikazane crvenom i narančastom bojom i informativne prijave (9) - prikazane crnom bojom.

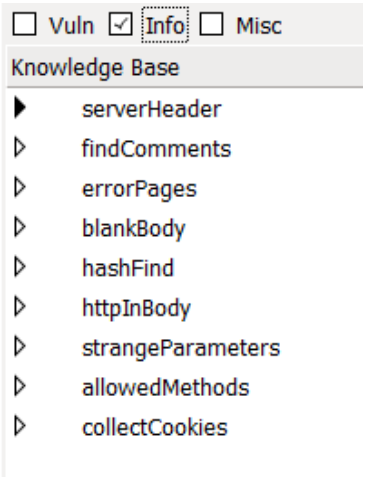
- Prijavljene ranjivosti:



Sl. 4.278. *Prikaz otkrivenih ranjivosti*

Na slici 4.28. su prikazane ranjivosti koje predstavljaju sigurnosni rizik. Crvenom bojom su označene ranjivosti s visokim stupnjem rizika (2), a narančastom bojom ranjivosti nižeg stupnja rizika (4).

- Prijavljene informativne prijave prikazane su na slici 4.29.

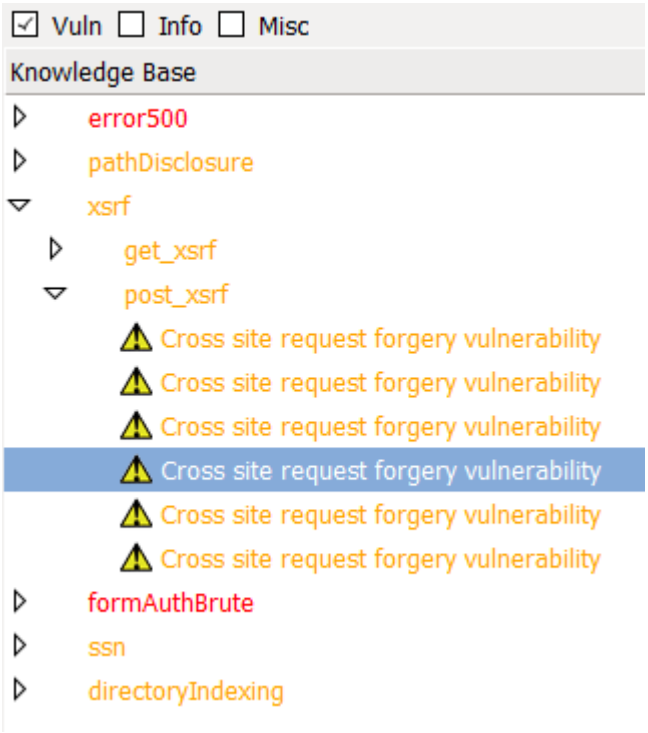


Sl. 4.289. *Prikaz otkrivenih upozorenja*

Osim ranjivosti korisnik može pregledati i upozorenja koja prema alatu ne predstavljaju veliki sigurnosni rizik, ali u svakom slučaju dobro ih je pregledati i popraviti ukoliko se radi o mogućim izvorima napada na aplikaciju.

4.4.4 Razgranato stablo prijavljenih w3af ranjivosti

Primjer razgranatog stabla ranjivosti prikazan je na slici 4.30.

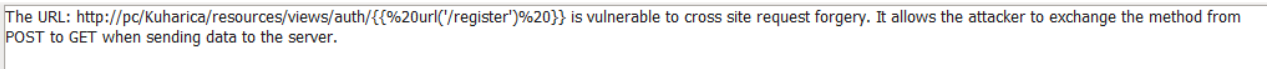


Sl. 4.30. *Razgranato stablo ranjivosti*

Slika 4.30. prikazuje razgranato stablo za xsrf ranjivost, tj. konkretnije za post_xsrf. Može se primijetiti da je uočeno četiri CSRF ranjivosti. Kao što je stablo razgranato za xsrf, isto tako može biti razgranato za bilo koju drugu uočenu ranjivost.

4.4.5 Opis ranjivosti prijavljene alatom w3af

Primjer opisa ranjivosti:



Sl. 4.291. *Opis ranjivosti*

U opisu ranjivosti navodi se URL adresa za koju je pronađena ranjivost te tip ranjivosti koji je pronađen (u ovom primjeru se radi o CSRF ranjivosti). Nakon navoda ranjivosti dan je kratki opis ranjivosti (CSRF može dopustiti napadaču mijenjanje POST zahtjeva u GET zahtjev pri slanju podataka poslužitelju).

4.5 Implementacija i analiza alata Nikto

Nikto je skener web poslužitelja koji je pisan u Pearl jeziku. Radi se o skeneru otvorenog koda koji je podržan na Windows, Mac i Linux operacijskim sustavima [37].

4.5.1 Preuzimanje i pokretanje Nikto skeniranja

Nikto alat za skeniranje web aplikacija je dostupan za preuzimanje na sljedećoj adresi: [49], a izvorni kod skenera se nalazi na sljedećoj adresi: [50]. Za početak testiranja potrebno je otvoriti komandnu liniju te unijeti adresu web stranice za koju se želi provesti testiranje na moguće ranjivosti.

Na slici 4.32 prikazan je primjer unosa adrese web stranice. Osim adrese postavljen je zahtjev za stvaranje html izvješća nakon testiranja. U ovom primjeru izvješće se sprema u report.html datoteku.

```
>nikto -o report.html -Format html -host http://test
```

Sl. 4.302. *Pokretanja testiranja*

4.5.2 Izvješće Nikto skeniranja prikazano u naredbenoj liniji

Nakon 8 minuta program je završio s testiranjem te su rezultati skeniranja pohranjeni u html datoteku „report.html“.

Izvođenje skeniranja prikazano je na slici 4.33.

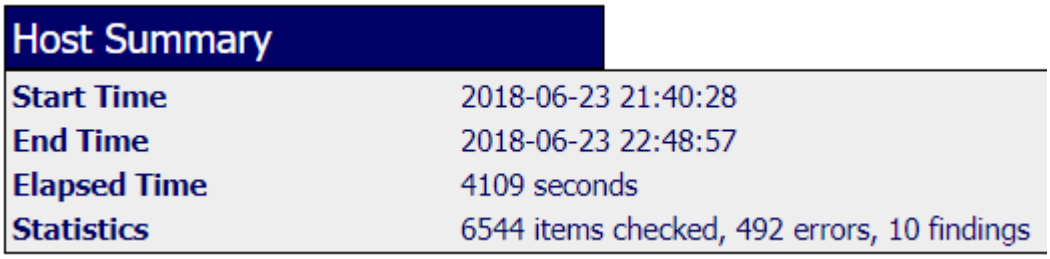
```
+ Start Time:         2018-06-23 21:40:28 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.21
+ Cookie XSRF-TOKEN created without the httponly flag
+ Retrieved x-powered-by header: PHP/5.6.21
+ The anti-clickjacking X-Frame-Options header is not present.
+ All CGI directories 'found', use '-C none' to test none
+ Server leaks inodes via ETags, header found with file /Kuharica/public/robots.txt, fields: 0x1a 0x54151e43d9fde
+ "robots.txt" retrieved but it does not contain any 'disallow' entries (which is odd).
+ OSVDB-5737: WebLogic may reveal its internal IP or hostname in the Location header. The value is "http://localhost/Kuharica/public./".
+ Allowed HTTP Methods: GET, HEAD, POST
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ OSVDB-3268: /Kuharica/public/images/: Directory indexing found.
+ OSVDB-3268: /Kuharica/public/images/?pattern=/etc/*&sort=name: Directory indexing found.
+ 6544 items checked: 492 error(s) and 10 item(s) reported on remote host
+ End Time:           2018-06-23 22:48:57 (GMT2) (4109 seconds)
```

Sl. 4.313. *Prikaz testiranja u naredbenoj liniji*

U naredbenoj liniji mogu se vidjeti podaci o početku i završetku skeniranja (datum i vrijeme) te druge značajke (podaci o poslužitelju, kolačiću i dr.).

4.5.3 Sažeti podaci o izvršenom Nikto skeniranju

Kratki podaci iz izvješća nakon završetka testiranja prikazani su na slici 4.34.



Host Summary	
Start Time	2018-06-23 21:40:28
End Time	2018-06-23 22:48:57
Elapsed Time	4109 seconds
Statistics	6544 items checked, 492 errors, 10 findings

Sl. 4.324. *Izvješće skeniranja pohranjeno u report.html datoteci, vrijeme i statistika*

U izvješću korisnik može pregledati kratke podatke o testiranju. Ti podatci uključuju datum početka i kraja testiranja, kao i vrijeme početka i kraja testiranja, proteklo vrijeme u sekundama te statističke podatke.

4.5.4 Prikaz ranjivosti u Nikto izvješću

Primjer prikaza ranjivosti prikazan je na slici 4.35.

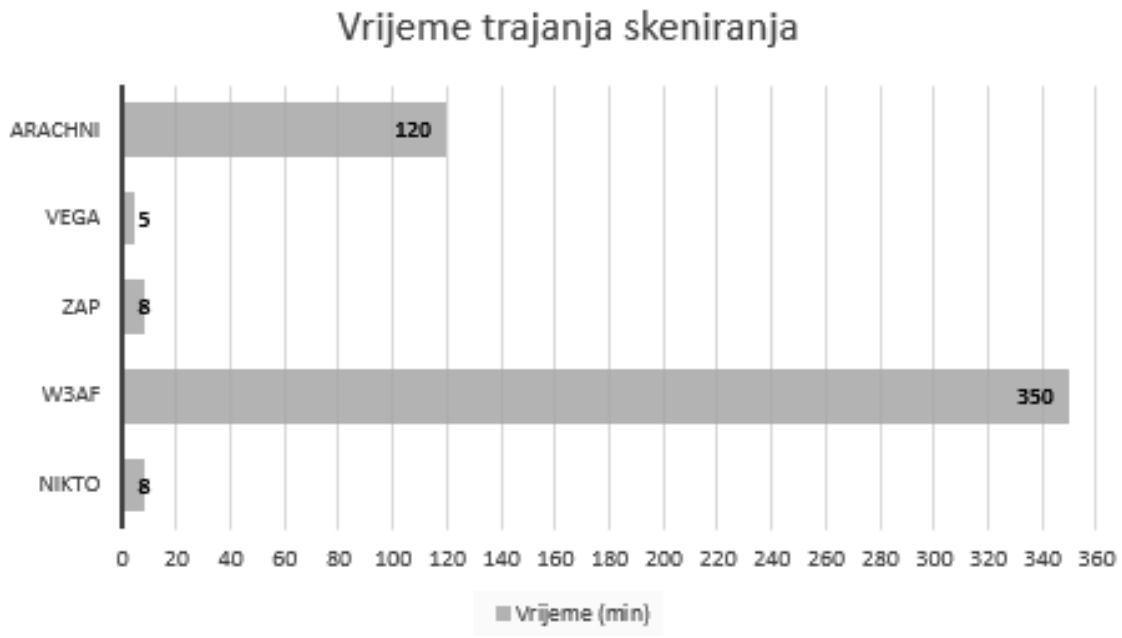
URI	/Kuharica/public/
HTTP Method	TRACE
Description	HTTP TRACE method is active, suggesting the host is vulnerable to XST
Test Links	http://pc:80/Kuharica/public/ http://192.168.1.11:80/Kuharica/public/
OSVDB Entries	OSVDB-877

Sl. 4.335. *Izvješće skeniranja pohranjeno u report.html datoteci, primjer prikaza ranjivosti*

Na slici 4.35. prikazan je primjer prikaza ranjivosti. Prikaz se sastoji od URI adrese, HTTP metode, opisa (u ovom primjeru radi se o HTTP TRACE metodi koja može upućivati na XST ranjivost), testnih poveznica i OSVDB ulaza.

4.6 Usporedba alata nakon testiranja

Na slici 4.36 su prikazana vremena (u minutama) potrebna za skeniranje aplikacije. Za neke alate je vrijeme automatski postavljeno, a za neke je bilo potrebno ručno odrediti vrijeme testiranja aplikacije (Arachni). Skeniranje je najduže trajalo kod alata w3af, a najmanje kod alata Vega.



Sl. 4 36. *Vremena trajanja skeniranja aplikacije*

U sljedećoj usporedbi naglasak je stavljen na tipove ranjivosti koji su pronađeni, a ne na ukupni broj pronalaska istih tipova ranjivosti. Na primjer: alat Arachni je pronašao CSRF tip ranjivosti koji se u aplikaciji pojavljuje na ukupno tri mjesta. Pri usporedbi, zbog jednostavnije analize, uzima se da je pronađen jedan tip ranjivosti (CSRF), a ne broj koliko puta se pojavljuje u aplikaciji.

Slika 4.37. prikazuje podatke o stupnjevima rizika različitih alata.

Sl. 4.37. *Stupnjevi rizika različitih alata*

Svaki alat je drugačije rađen i zbog toga izvješća imaju drugačiji izgled. Kod većine alata (Arachni, Vega, Zap) su tipovi rizika podijeljeni na visoki, srednji i niski stupanj rizika te informativne prijave. W3af prijava pokazuje samo dva stupnja ranjivosti (visoki i niski) te informativne prijave, dok kod izvješća Nikto skenera piše da je pronađene deset tipova ranjivosti (ne piše kojeg su stupnja ranjivosti).

Na slici 4.38 grafički je prikazana usporedba ukupnog broja pronađenih različitih tipova ranjivosti i upozorenja po alatima.

Sl. 4.38. *Ukupan broj ranjivosti i upozorenja*

Broj pronađenih ranjivosti razlikuje se od alata do alata, kao što se može vidjeti na slici 4.38. Prema dobivenim rezultatima najviše tipova ranjivosti je pronašao alat w3af, a najmanje Arachni alat. Ne možemo sa sigurnošću utvrditi da je w3af najbolji alat zato što je pronašao najveći broj ranjivosti jer u obzir trebamo uzeti i moguće netočno pozitivne prijave (engl. false positive). Alat Arachni koji je prijavio najmanji broj različitih tipova ranjivosti može biti najbolji, ali i ne mora zato je uz automatizirano skeniranje potrebno izvršiti i ručno testiranje.

Alat w3af koji je imao najduže vrijeme skeniranja prijavio je najveći broj informativnog sadržaja te najveći ukupni broj svih tipova upozorenja. Alat Vega koji je imao najkraće vrijeme skeniranja prijavio je drugi najveći broj svih tipova upozorenja, te najveći broj ranjivosti visokog i srednjeg stupnja rizika. Nikto i ZAP imaju jednako vrijeme skeniranja i jednak ukupni broj prijava svih

tipova ranjivosti. Osim toga alat ZAP ima najveći broj prijava niskog stupnja ranjivosti, a alat Arachni ima najmanji ukupni broj prijava svih tipova ranjivosti.

Iako se izvješća nakon skeniranja razlikuju u načinu prikaza podataka i broju pronađenih grešaka i ranjivosti, kao i prikazu i nazivima ranjivosti, alati su pronašli i ukazuju na dosta istih tipova ranjivosti. Nakon analize alata i izvješća možemo utvrditi da je puno jednostavnije aplikaciju testirati uz pomoć automatiziranog skenera, ali uz skener potreban je ljudski nadzor i logika te ručno testiranja kako bi se pronašao i otklonio najveći broj sigurnosnih rizika.

5 ZAKLJUČAK

Prije same analize alata bilo se potrebno upoznati sa sigurnosnim propustima koji se često pojavljuju. Lista najčešćih deset ranjivosti predstavlja najkritičnije sigurnosne propuste koji su se pojavljivali u web aplikacijama i na koje je potrebno obratiti pažnju pri testiranju kako bi se zaštitila aplikacija i osigurala sigurnost korisničkih podataka. Za dobro razumijevanje ranjivosti osim navoda i kratkog opisa, potrebno je proučiti primjere napada kao i rizike koje ranjivost uzrokuje te savjete za sprječavanje.

Nakon proučavanja ranjivosti prelazi se na analiziranje pet alata za skeniranje web aplikacija u potrazi za mogućim sigurnosnim propustima. Prije korištenja alata, kako bi osigurali najbolje performanse i rezultate testiranja, neophodno je upoznati se s načinom na koji su alati izgrađeni i funkcioniraju. U ovom radu su istraženi sljedeći alati: Arachni, Vega, OWASP ZAP, w3af i Nikto. Radi se o alatima besplatnima za korištenje (osim Arachni alata ukoliko se koristi u komercijalne svrhe) s dostupnim izvornim kodom. Svi alati dostupni su za sljedeće operacijske sustave: MS Windows, Linux i Mac OS.

Kako bi skeniranje bilo moguće potrebno je pronaći javno dostupnu aplikaciju za skeniranje ili razviti vlastitu aplikaciju na kojoj će se moći testirati alati. Alati su jednostavni za korištenje, a nakon unosa URL-a aplikacije i skeniranja dobije se izvještaj o potencijalnim ranjivostima u različitim formatima. Svaki od alata ima svoje pojedinosti kojima se razlikuje od drugih alata. S obzirom na potrebe, korisnik bi trebao odabrati odgovarajuće alate za provjeru pogrešaka kako bi se otkrio veći broj sigurnosnih propusta. Osim automatiziranih alata koji uvelike mogu pomoći u testiranju i otklanjanju ranjivost, potrebno je izvršiti i ručna testiranja jer se pojedine ranjivosti teško otkrivaju automatiziranim skenerima zbog velike kompleksnosti i dinamičnosti web aplikacija, a logičkim razmišljanjem mogu se lako uočiti pri ručnom testiranju. Kako bi se osigurala što veća pouzdanost i kvaliteta aplikacija testeri i programeri trebaju biti svjesni postojanja ranjivosti i mogućih šteta te obratiti pozornost na preporuke za sprječavanje napada. Programeri zainteresirani za sigurnost mogu dati svoj doprinos i uključiti se u poboljšanje već postojećih alata. Mnogi skeneri se mogu proširiti preko modula, a osim otvorenog koda postoje pojašnjenja pojedinih dijelova koda koja mogu pomoći u razumijevanju funkcioniranja alata te dati ideje za poboljšanje i daljnji razvoj alata za skeniranje web aplikacija.

LITERATURA

[1] IBM: OWASP top 10 vulnerabilities, dostupno na: https://www.ibm.com/developerworks/library/se-owasptop10/, 10.6.2018.

[2] OWASP: Testing for LDAP Injection , dostupno na: https://www.owasp.org/index.php/Testing_for_LDAP_Injection_(OTG-INPVAL-006), 18.3.2018.

[3] OWASP Top 10-2017, dostupno na: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf, 21.5.2018.

[4] S. Faust, SPI Labs: LDAP Injection, dostupno na: http://www.networkdls.com/articles/ldapinjection.pdf, 15.2.2018.

[5] R. Alnaqeib, F. H. Alshammari, M. A. Zaidan, A. A. Zaidan, B. B. Zaidan, Z. M. Hazza, An Overview: Extensible Markup Language Technology, Journal of computing, br. 6, sv. 2, lipanj 2010.

[6] w3schools: DTD, dostupno na: https://www.w3schools.com/xml/xml_dtd_intro.asp, 16.2.2018.

[7] acunetix: What is XML External Entity (XXE)?, dostupno na: https://www.acunetix.com/blog/articles/xml-external-entity-xxe-vulnerabilities/, 19.2.2018.

[8] OWASP: XML External Entity (XXE) Processing, dostupno na: https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing, 7.3.2018.

[9] Hack2Secure: Understanding Broken Access Control Risk, dostupno na: https://www.hack2secure.com/blogs/understanding-broken-access-control-ris, 10.3.2018.

[10] acunetix: What is Insecure Deserialization?, dostupno na: https://www.acunetix.com/blog/articles/what-is-insecure-deserialization/, 12.3.2018.

[11] Spring Web Flow, dostupno na: https://projects.spring.io/spring-webflow/, 15.3.2018.

[12] Gotham Digital Science: CVE-2017-4971, dostupno na: https://blog.gdssecurity.com/labs/2017/7/17/cve-2017-4971-remote-code-execution-vulnerability-in-the-spr.html, 16.3.2018.

[13] Flexera: CVE-2017-5638, dostupno na https://blogs.flexera.com/sca/2017/09/an-analysis-of-the-apache-struts-2-vulnerability/, 16.3.2018.

[14] Imperva: CVE-2017-5638, dostupno na: https://www.imperva.com/blog/2017/03/cve-2017-5638-new-remote-code-execution-rce-vulnerability-in-apache-struts-2/, 16.3.2018.

[15] Hack2Secure: Insufficient Logging And Monitoring A Brief Walk Through, dostupno na: https://www.hack2secure.com/blogs/insufficient-logging-and-monitoring--a-brief-walk-through, 18.3.2018.

[16] Imperva: The State of Web Application Vulnerabilities in 2017, dostupno na: https://www.imperva.com/blog/2017/12/the-state-of-web-application-vulnerabilities-in-2017/, 19.3.2018.

[17] OWASP Top Ten Project, dostupno na: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, 15.6.2018.

[18] Arachni Scanner, dostupno na: http://arachni-scanner.com/, 25.6.2018.

[19] Infosec Institute: Web Application Testing with Arachni, dostupno na: https://resources.infosecinstitute.com/web-application-testing-with-arachni/#gref, 25.6.2018.

[20] M. McPhee, Mastering Kali Linux for Web Penetration Testing, Packt, 2017

[21] Difference Between, dostupno na: http://www.differencebetween.net/technology/difference-between-cvs-and-svn/, 20.6.2018.

[22] Subgraph: Vega, dostupno na: https://subgraph.com/vega/documentation/index.en.html, 30.6.2018.

[23] Wikipedia: TLS, dostupno na: https://hr.wikipedia.org/wiki/TLS, 21.6.2018.

[24] Knoldus: How to start with Vega : The web security scanner? , dostupno na: https://blog.knoldus.com/start-vega-web-security-scanner, 23.6.2018.

[25] GitHub, Subgraph: Vega, dostupno na: https://github.com/subgraph/Vega/wiki/Basic-Modules, 24.6.2018.

[26] OWASP Zed Attack Proxy Project, dostupno na: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project, 18.7.2018.

[27] WhatIs.com, dostupno na: ttps://whatis.techtarget.com/definition/spider, 18.7.2018.

[28] G. T.-y, S. Y.-s., F. Y.-y., Research on Software Security Testing, World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering, br. 9, sv. 4, 2010.

[29] techopedia: Fuzz Testing, dostupno na: https://www.techopedia.com/definition/13625/fuzz-testing, 18.7.2018.

[30] Infosec Institute: Introduction to OWASP ZAP for Web Application Security Assessments, dostupno na: https://resources.infosecinstitute.com/introduction-owasp-zap-web-application-security-assessments/#gref, 19.7.2018.

[31] CyberSecology: The OWASP Zed Attack Proxy (ZAP) Scanner, dostupno na: http://cybersecology.com/the-owasp-zed-attack-proxy-zap-scanner/, 20.7.2018.

[32] Multi-step scanning in ZAP, dostupno na: http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6823/pdf/imm6823.pdf, 20.7.2018.

[33] ZAPpingTheTop10, dostupno na: https://www.owasp.org/index.php/ZAPpingTheTop10, 21.7.2018.

[34] w3af, dostupno na: http://w3af.org/, 27.7.2018.

[35] w3af DOCS, dostupno na: http://docs.w3af.org/en/stable/install.html, 27.7.2018.

[36] Cirt.net, dostupno na: https://cirt.net/Nikto2, 29.7.2018.

[37] Nikto: The Manual, dostupno na: https://cirt.net/nikto2-docs/, 29.7.2018.

[38] SourceForge: Nikto QT, dostupno na: https://sourceforge.net/projects/niktoqt/, 29.7.2018.

[39] MadIrish: Using the Nikto Web Application Vulnerability Scanner, dostupno na: https://www.madirish.net/547, 30.7.2018.

[40] Unixmen: How To Install Nikto Web Scanner To Check Vulnerabilities, dostupno na: https://www.unixmen.com/install-nikto-web-scanner-check-vulnerabilities/, 28.7.2018.

[41] Arachni: Download, dostupno na: http://www.arachni-scanner.com/download/, 15.1.2018.

[42] GitHub: Arachni, dostupno na: https://github.com/Arachni/arachni, 20.6.2018.

[43] Subgraph, Vega: Download, dostupno na: https://subgraph.com/vega/download/, 16.1.2018.

[44] GitHub: Subgraph, Vega, dostupno na: https://github.com/subgraph/Vega, 24.6.2018.

[45] zaproxy: Downloads, dostupno na: https://github.com/zaproxy/zaproxy/wiki/Downloads, 4.2.2018.

[46] GitHub: zaproxy, dostupno na: https://github.com/zaproxy/zaproxy, 4.7.2018.

[47] sourceforge, w3af: Download, dostupno na: https://sourceforge.net/projects/w3af/, 7.2.2018.

[48] GitHub: w3af, dostupno na: https://github.com/andresriancho/w3af, 5.7.2018.

[49] Nikto2: Download, dostupno na: https://cirt.net/Nikto2, 10.2.2018.

[50] GitHub: Nikto, dostupno na: https://github.com/sullo/nikto, 21.6.2018.

KRATICE

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
BSD	Berkeley Software Distribution
CLI	Command line interface
CORS	Cross-Origin Resource Sharing
CRLF	Carriage Return Line Feed
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CVS	Concurrent Versions System
DAST	Dynamic application security testing
DOM	Document Object Model
DoS	Denial of Service
DTD	Document Type Definition
EPL	Eclipse Public License
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
ID	Identifier
IP	Internet Protocol
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
LFI	Local File Inclusion
MVC	Model-View-Controller
NBE	nBill Extension
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
PXSS	Prevent Cross Site Scripting
RCE	Remote Code Execution
RFI	Remote File Inclusion
RPC	Rich Client Platform
RXSS	Reflected Cross Site Scripting
SQL	Structured Query Language

SS	Server Side
SSH	Secure Shell
SSI	Server-Side Includes
SSJS	Server Side JavaScript
SSL	Secure Sockets Layer
SSRF	Server-side Request Forgery
SVN	SubVersioN
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3af	Web Application Attack and Audit Framework
WAF	Web application firewalls
WSDL	Web Services Description Language
XML	EXtensible Markup Language
XSD	XML Schema Definition
XSS	Cross-site scripting
XXE	XML External Entity
ZAP	Zed Attack Proxy

SAŽETAK

Svaka programska podrška trebala bi biti sigurna za korištenje i čuvati osjetljive korisničke podatke. Zbog velikog broja upada u sustav i krađe podataka, potrebno je razmotriti oblike ranjivosti sustava i obratiti pažnju na moguće rizike napada i preporuke za razvoj sigurne programske podrške. Nakon razvoja, potrebno je provesti detaljna testiranja kako bi se osigurala što veća kvaliteta i pouzdanost programske podrške. Osim ručnih testiranja, postoji veliki broj automatiziranih alata koji skeniraju aplikaciju te daju izvještaj potencijalnih ranjivosti u raznim formatima. Neki od alata otvorenog koda koji se koriste za skeniranje aplikacija su Arachni, Vega, OWASP ZAP, w3af i Nikto. Dostupni su za MS Windows, Linux i Mac OS. Pokrivaju širok spektar ranjivosti i sigurnosnih propusta u internetskim aplikacijama. Otkrivanje im je nekih ranjivosti zajedničko, a za neke je ranjivosti potrebno proučiti specifikacije i odabrati alat koji je specijaliziran za tu vrstu ranjivosti. Alati otvorenog koda omogućuju korisnicima dodavanje vlastitih testova preko modula i drugih dodataka. Zbog stalnog pojavljivanja novih ranjivosti, treba biti u toku kako s ranjivostima, tako i s novim značajkama i modulima alata. Oni koji žele dati svoj doprinos u poboljšanju sigurnosti, umjesto postavljanja temelja novog alata, mogu se uključiti u razvoj i poboljšanje nekog već postojećeg alata.

Ključne riječi: ranjivost, sigurnosni propust, skener, testiranje, web aplikacija.

ABSTRACT

Usability analysis of software security tools

All software should be safe for using and protecting sensitive user data. Due to a large number of system intrusions and data theft, it is essential to consider different types of system vulnerabilities and pay attention to potential attack risks, as well as recommendations of the secure software development. After a development phase, it is necessary to conduct detailed testing to ensure high quality and reliability software. In addition to manual testing, many automated security tools, which scan the application and provide a report of potential vulnerabilities in various formats, are available for the usage. Some open source tools which are used for scanning applications are Arachni, Vega, OWASP ZAP, w3af and Nikto. They are available for MS Windows, Linux and Mac OS. Furthermore, they cover many vulnerabilities and security flaws in web applications. On the one hand, their communal feature is detection of some vulnerability types, while on the other hand, in the case of some specific vulnerability, it is important to study specifications and choose the most appropriate tool that is specialized for the target type of vulnerability. Open source tools allow users to add their own tests through modules and other plug-ins. Due to the continuous emergence of new vulnerabilities, it is important to stay up to date with both vulnerabilities as well as new features and modules of scanning tools. Those who want to contribute to the community and security improvements, instead of developing a new tool, can involve themselves in the development and improvement of some existing tools.

Keywords: vulnerability, security flaw, scanner, testing, web application.

ŽIVOTOPIS

Kristina Javorek je rođena 01.07.1994. u Slavonskom Brodu gdje je odrasla i završila matematičku gimnaziju. Školovanje nastavlja u Osijeku na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Završava preddiplomski studij s prosjekom 4.906 čime stiče naziv Sveučilišna prvostupnica inženjerka računarstva. Stručno osposobljavanje obogaćuje sudjelovanjem na Span i Inchoo akademijama te međunarodnom studentskom projektu TeamSoc21, gdje stiče nova znanja i vještine. Preko Erasmus programa, praksu je odradila u Njemačkoj firmi aaronProjects GmbH. 2016. god. dobiva Priznanje fakulteta za uspješnost u studiranju, a 2018. god. dobitnica je Rektorove nagrade za izvrstan seminarski rad. Uz materinski hrvatski jezik, govori engleski (B2/C1) i njemački (B2). Od certifikata se mogu istaknuti: HTML5 Application Development Fundamentals, Principles of Machine Learning i Dana Science Essentials, a od projekata: Brain game (Igra pamćenja, C#), Feritgal (Galerija fotografija, PHP, HTML, CSS), Robot koji izbjegava objekte (Arduino Uno), MealPlanner (Android) te Klasifikacija slika Pas/Mačka (Strojno učene).

PRILOZI (na DVD-u)

Prilog 1: Elektronička verzija diplomskog rada (.doc)

Prilog 2: Elektronička verzija diplomskog rada (.pdf)

Prilog 3: Izvještaji skeniranja